A QUANTUM COMPUTATIONAL LINGUISTIC MODEL

BEN SPROTT

Abstract. Described here is a quantum computational linguistic model. It is based on the fact that formal grammars have representations in the operators on a Hilbert space. These operators form the set of unitary transformations on an n level quantum system. A simplified version of English is cast as a group and basic properties of that language are probed algebraically. It is shown that unitary operators will not provide a sufficient space for a representation of a realistic, natural language. The space of Superoperators is suggested as the next step.

There are no calculations of the matrices which act as representations of any algebraic structure which may model natural language. Instead the paper simply gathers together some disparate areas of mathematics and linguistics so that, maybe, a few practitioners of those areas might then actually provide the final representations.

1. Introduction

How does one model a natural language and, once devised, what is the use of these models? While automata theory and the theory of formal grammars may be a weak place to start discussing the finer details of natural language, computational linguistic models are nevertheless an interesting discourse. A computational linguistic model can be used to parse natural languages, i.e. deciding whether or not a given sentence is a valid sentence in a given language. It can also provide a mathematical framework in which to discuss the structure of a language and perhaps even the reasons why certain aspects of a language are the way they are. This is true of the quantum computation linguistic model.

The quantum computer is distinctly different from the one you are using now. Simply put, a quantum computer works on quantum principles. The idea of a quantum computer was first considered several decades ago. Since then, much attention has been paid to understand the strengths, weaknesses, similarities and differences between the quantum and classical computers.

Quantum systems like particle scattering experiments, photon polarization, and manipulation of nuclear spins could be used to build a quantum computer. In this paper, one will find only the most abstract formalisms for talking about computers. By using a very abstract language, it is easy to see why quantum computing, as a framework, is an excellent place to talk about linguistic models. Also, in this more abstract language, it is easy to see just how one would go about implementing these algorithms.

2. Formal Languages and Grammars

2.1. **Modeling Natural Language with Formal Languages.** Though formal languages, those found in abstract computer science, are very weak at encapsulating

natural language, let us begin here and see if we can't build up into something complex. Perhaps, the more complex structures we find later will be more suitable for modeling natural language.

First, there is the alphabet. The alphabet consists of a set of distinguishable symbols $\Sigma = \{A, B, C \ldots, NP, VP, Det \ldots\}$. You might already have noticed the symbols NP and VP in there already. Some might be balking at my hesitance to simply jump to a grammar with its definition of terminals and non terminals, but lets indulge the non linguist for now. Given this alphabet, a first attempt at a language may be a set of strings in these symbols.

Here is a very simple language. The alphabet is

$$\Sigma = \{1\}$$

and the language is the set of all strings of 1s, also given by Σ^*. How does this language work? Well, it works by attaching distinguishable concepts to distinguishable symbols. For example,

$$\begin{aligned} 1 &= \textit{Dog} \\ 11 &= \textit{Cat} \\ 111 &= \textit{I need food} \\ 1111 &= \textit{There is a shocking lack of decorum in this establishment}! \\ &\ldots \end{aligned}$$

This would be a very bad language as it would have a serious memory load problem. Regardless, it has the basic structure which we need to probe what a language is. In this case, all possible strings are valid elements in the language.

Let's try another example. Let's stick with the same alphabet, but we will redefine our language as the set of all even strings so we end up with something like

$$\begin{aligned} 11 &= \textit{Dog} \\ 1111 &= \textit{Cat} \\ 111111 &= \textit{I need food} \\ 11111111 &= \textit{There is a shocking lack of decorum in this establishment!} \\ &\ldots \end{aligned}$$

Now we find the case that some words, like 1, are not valid elements of the language because they are not even in length.

Let us apply this analogy to English. The following sentence is a valid English sentence

> My name is Adam.

While this sentence is not

> Name my Adam is.

Why does this second sentence sound so bad? Who knows! Chomsky might have suggested that inside our heads is a little algebraic machine that churns out sentences with a specific algebraic structure. Answering this question really isn't the point of a linguistic model. The model is simply there to see how well, and how efficiently, one may define a set of sentences which map accurately onto that which humans would say constitutes their language.

2.2. **Formal Grammars.** A formal Grammar is very much like the terrible languages defined above. The difference is in the delicate rules for what constitutes a valid sentence. Chomsky's formal grammars go one step further by breaking the symbols of the language into two layers.

The first layer, called the *non-terminals*, is a grammatical layer in which we find symbols like NP. NP stands for *noun phrase* and is meant to represent, on a deeper level, the concept of the noun. The algebraic structure, rather than being between the words, is between the elements of this abstract, grammatical layer.

On top of this is the layer of words, called terminal symbols. In here we find familiar words like *the* and *dog*. The idea being that, once the strings of grammatical concepts are laid out in the form of *non-terminals*, they can be replaced by anything in the group of words they relate to.

Considering the following two English sentences

> The girl I love is dead

and

> Girl the is I love dead

By commuting i.e. switching the order of *the* and *girl*, we have taken something that feels like a good English sentence and made it into something that definitely does not feel like a good sentence. Thus there is an algebraic analogy and these algebraic structures have representations as groups and formal grammars. In other words, we can model these languages (well or poorly) using formal grammars and groups.

In the case of groups representations of a language, every sentence which belongs to the grammar, i.e. is a good sentence, belongs to the identity equivalence class of the group used to represent the grammar. I will return to groups in detail in a moment.

Defining a formal grammar is similar to defining a formal language in that there is an alphabet, and a set of rules for defining what is a valid word or sentences. A formal grammar is thus given as $\Gamma = \{ \text{V}, \text{A}, \pi, \sigma \}$. The finite set of symbols V, called the vocabulary, is composed both of what are called terminal symbols as well as non-terminal symbols. The terminal symbols can be identified with words, while the non terminal symbols are parts of the grammatical structure like NOUN PHRASES, and VERB PHRASES.

If, at some point, while building a sentence in Γ you find a non terminal in the sentence, then the sentence is not yet complete. If you find only terminal symbols, then the sentence cannot be changed further and is thus complete. When the sentence is complete, and there are only non-terminal symbols left, it should look as familiar as anything you might find in your favorite publication.

The nonempty subset $A \subset V$ is composed of only terminal symbols. π is a set of what are called productions which are identifications between words in V. These are the rules of the grammar and are most important. These rules define what makes a valid sentence, and how to build sentences. They are equivalent to the relators in a finitely presented group. A typical production might be $ab \to cd$ and indicates that ab may be replaced by cd. They are equal so let us write that, $ab = cd$. This kind of equation would be the relator in a finitely presented group.

π is a finite subset of $(V/A)^+ \times V^*$. $\sigma \in \text{V/A}$.

As an example, let R be a non terminal symbol, and let a, b be terminal symbols. Thus, V={R,a,b }, A={ a,b }, σ=R and here is the set of productions

$$\underline{\pi} \tag{2.1}$$

$$\mathrm{R} \rightarrow \mathrm{aRb} \tag{2.2}$$

$$\mathrm{R} \rightarrow \epsilon \tag{2.3}$$

ϵ is the empty string. We begin constructing words by first writing down σ and then whatever we find that is equal to that, we may substitute, so $\sigma \rightarrow$R. We can then do transformations based on what R is identified with. A typical chain of productions would work as
$\sigma \rightarrow \mathrm{R} \rightarrow \mathrm{aRb} \rightarrow \mathrm{aaRbb} \rightarrow \mathrm{aaaRbbb} \rightarrow \mathrm{aaabbb}$
where the last transition was performed using the relation $\mathrm{R} \rightarrow \epsilon$. It should be apparent that this grammar constructs the familiar language

$$L_{a^n b^n} = \{a^n b^n | n \in \mathbb{Z}, n > 0\} \tag{2.4}$$

from equation (**??**). The language derived from a grammar Γ is called L(Γ).

3. Quantum Computers

Quantum computing is different from classical computing in many respects. A quick, though abstract way to point out the differences is to simply consider two computers. Both computers have a countable set of distinguishable, or orthogonal internal states. Let the quantum and classical computers have states Q, C respectively. At any given moment, the state of the classical computer is $c \in C$ and transitions can be made from state to state.

The set of all transitions between internal states of the classical computer is simply the set of maps from a finite set to itself, $M : C \rightarrow C$. The single difference between the classical and the quantum computer, is that there exists a *continuous* and reversible transformation between any of the pure states of the quantum computer. A pure state is one for which the observer has exact knowledge and is able to perform an experiment for which the outcome can be predicted with certainty. The set of transformations is a compact Lie group. The set of pure states of the quantum computer is described by an n dimensional Hilbert space $\mathfrak{H}_n$. Take a basis for this Hilbert space as $|q_i\rangle \in Q$. Any pure state is a complex linear superposition of these states

$$|\psi\rangle = \sum_i \alpha_i |q_i\rangle \tag{3.1}$$

The computer should be set to some initial state,

$$|q_0 >, c_0$$

In terms of parsing a natural language, we need to identify each of the words in a language with a transformation of the machine. In the case of the classical computer, we identify words with maps from some finite set to itself. In the quantum case we identify words with unitary transformations.

The sentence is then a concatenation of maps chosen from an appropriate set. If the total map sends the computer to a given stateq_{acc} then the sentence is a valid sentence. If the map sends the computer to any other state, then it is not a valid sentence. The computer, or machine, is another way of defining the language since it will always tell you if a given sentence is or is not in the language.

The computer will ***not*** understand anything, nor will it offer us a means of grasping the semantics or meaning of a language. It is a model which is meant only to encapsulate valid and invalid sentences. In other words, it is a model of the theory of the language. Say one were to give axioms for an abstract algebraic structure which identified all valid English sentences. This would also be a model of the theory of the language.

We now leave behind the classical computer as it was just used as a means of understanding the quantum computer. This is the basic model of the computer which is at the heart of the QCLM. The Hilbert space is chosen to be of a size which suits the language we are trying to model. Once a suitable algebraic structure is chosen to describe the language in question, the task remains to find representations of that structure within SU(n). To get a handle on this process, let's look at finitely presented groups. Finitely presented groups have represnentations in the space of operators on SU(n) and they themselves can be used to find representations of formal grammars.

Later on it will be shown that a better approximation to a natural language can only be found using representations within the Superoperators, rather than in the unitary transformations.

4. Group Theory

Group theory is a subject in abstract algebra, where the sets of interest are endowed with very specific algebraic properties. Groups are used in many branches of science since the generality of these structures allows for a wide range of models.

An abstract algebra must have a binary operator that acts on two elements of the set, familiar examples being addition and multiplication. A group G is a set of elements that has at least one binary operation, let us call it $*$ for now, and the following properties

- $\exists\ \mathbb{I} \in G$ s.t. $\mathbb{I} * a = a * \mathbb{I} = a$, $\forall\ a \in G$ (existence of identity)
- $a * b = c \in G$, $\forall\ a, b \in G$ (closure)
- $\forall a \in G$, $\exists\ a^{-1}$ s.t. $a * a^{-1} = a^{-1} * a = \mathbb{I}$ (inverse)
- $(a * b) * c = a * (b * c)$ (associativity)

From this point on, binary operations will not be identified with any special symbol like $*$. Instead, concatenation of elements of a group will implicitly assume a binary operation between them.

Abstract algebra is so named, as it deals with the simplest set of rules governing a mathematical object. These rules may apply to a whole range of mathematical objects. In the case of quantum mechanics, since we describe our states as vectors, the natural objects that would act on them are matrices. Many algebraic structures of interest have what are called matrix representations. This, of course, is true of formal grammars, and hopefully even natural languages.

4.1. Finitely Presented Groups.

Given a set of symbols in an alphabet Σ, we can identify them with generators of a group. Combine the generators as you would letters to form words. Combining the letters this way naturally assumes the existence of some binary operation. The free group is the set of all combinations of the generators and their inverses. It will be denoted by $\mathbb{G} = \Sigma^* * (\Sigma^*)^{-1}$ which is a set that includes the identity I. Notice that any combination of words in $\mathbb{G}$ is in Σ^*. If we let each symbol have an inverse, then every word has an inverse by

simply reversing the order of the letters and changing all letters to their inverses

$$\begin{aligned} x &= abcd \\ x^{-1} &= d^{-1}c^{-1}b^{-1}a^{-1} \\ xx^{-1} &= abcdd^{-1}c^{-1}b^{-1}a^{-1} = I \end{aligned}$$

Thus every element of the group has an inverse. The free group is not particularly interesting. For example, if the alphabet is just $\Sigma = 1$, and the binary operation is addition, then Σ^* would be isomorphic to $\mathbb{Z}$, the integers.

The interesting groups are formed when added structure is enforced by what are called relaters. A relater is any word in the alphabet that is equivalent to the identity. Any relater can be fathomed. For example, given $\Sigma = \{a, b\}$ one may wish to have it that a commutes with b, and thus $ab = ba$. We can rewrite that word and receive the equivalent expression $aba^{-1}b^{-1} = \mathbb{I}$ which is a relater. Along with the generators a, b, we now have an entirely new algebraic structure, namely the infinite abelian group with two generators.

With the generators and the relaters, we can form what is called a presentation of a group. Groups defined by generators and relaters are presented in a the following fashion

$$\{a, b, c, ...m|abc = I, ...\}$$

i.e.

$$\{\Sigma|Relator1, Relator2, \ldots\}$$

The presentation of $\mathbb{Z}_2$ is

$$\{b|bb\}$$

We concern ourselves here with finitely presented groups, i.e., groups having a finite set of generators and relaters.

The notation $\langle|\rangle$ is standard in combinatorial group theory but it is too easily confused with Dirac Bra Ket notation. For this reason we use the standard set notation $\{|\}$.

No relater can be equal to some other relater by insertion of other relaters. As an example, consider the group $\{a, b|a^2, b^8\}$, and the word $ab^4a^2b^4a = \mathbb{I}$. Words of this type are called trivial relaters and their inclusion does not change the group structure. The word $ab^4a^2b^4a$ is trivial as it is formed by the insertion of $b^8 = \mathbb{I}$ into $a^2 = \mathbb{I}$, and the insertion of a^2 into the resulting word. We can see here that the group structure creates a natural equivalence relation on the free group. If x and y are words in G and x can be turned into y by inserting and removing various identity words, then $x =_G y$. Given any word y, there is an equivalence class $[y]$ associated with that word. To determine if a word is in the same equivalence class as the identity is called the word problem. The word problem is, actually a very interesting one and will be discussed in greater detail below.

The presentation of a group G offers an equivalence relation $=_G$ on the set of all words $\mathbb{G}$. Two words are equivalent according to $=_G$ if one can transform one word into another using the relaters. Recall that the relaters are all equal to the identity, thus insertion of those words, deletion of those words, or concatenation with those words has no effect on the group element. For every element $w \in G$, there is an equivalence class $[w]$ and every word in $[w]$ is equivalent relative to $=_G$.

As an example, consider the group

$$G = \{a, b | a^2\}$$

It should be clear that a^2 is equal to the identity, thus $a^2 \in [I]$. The following statements are also true

$$a^{2n} \in [I]$$
$$aab \in [b].$$

5. Grammars as Groups

Here is a very simplistic generative grammar which we take as a very simple model of the English language. The productions are

$$\begin{aligned} S &\rightarrow NPVP \\ VP &\rightarrow VNPPP \end{aligned}$$

We can present this algebraic structure a little differently. In terms of a finitely presented group, the set of sentences S is replaced by the equivalence class of the identity in some group. To say that $S \rightarrow NPVP$ is to say that $NPVP = I$. This equation is a familiar relator and helps define the algebraic structure. Algebraically speaking, $NP = VP^{-1}$

$$G = \{NP, VP, V, PP | I = NPVP, VP = VNPPP\} \tag{5.1}$$

One exciting thing to be seen right away is that NP and VP commute. This means that if NP VP is a valid sentence, so is VP NP. When compared to English, we see right away that this is true in some cases. Take, for example, the following sentences

i am

and

am i

They are both valid. Now, this rule has limited applicability which is to say that this group is far too simplistic to thoroughly model language. I will come back to this in greater detail in section 8.

6. Representations in SU(n)

We can now start talking about the quantum computational linguistic model. Recall that the model is just a representation of some algebraic structure. The algebraic structure is a representation of some formal grammar. The formal grammar is a model of natural language, but a poor one. So let's begin.

This part is quite simple. We just have to find representations of the group 5.1 in the unitary operators on a Hilbert space.

What's funny is that unitary matrices are a representation of the operators on a Hilbert space. Where does this chain of representations end! To answer that question I say this: Not with me!

Unfortunately, I am not going to provide the matrices which act as representations of that group 5.1 or any other group which may better model natural language. The point of this paper was not to provide an actual model of natural language.

Instead it just gathers together some disparate areas of mathematics and linguistics so that, maybe, a few practitioners of those areas might then actually provide the final representations. I just hope I am making things clear enough so that those people can forge ahead and get the work done under the belief that this path really does exist.

7. Better Models for More Natural Languages

In this section, we see the power of the abstract modeling techniques really coming together to get a grip on real natural language. The hope is that, within the Superoperators, there is a representation of a probabilistic context-free phrase structure grammar. These are used by computational linguists today (See for example

Read the following two sentences.

i am
am i

Both of these feel like good sentences. Looking at their structures we see that we can form these sentences by the following productions

$$\begin{aligned} S &\rightarrow NPVP \\ NPVP &\rightarrow \text{i am} \\ VPNP &\rightarrow \text{am i} \end{aligned}$$

Thus we see that the nonterminals NP and VP commute. Regardless of the fact that VP NP seems to be reserved for questions, they are still valid sentences. However, let's consider another example.

she ran to the house
ran she to the house

Now the second one feels a bit strange. It feels like a question and for some reason, it feels best if imagined spoken by an ancient British king! Finally, consider these sentences.

the dog ran
ran the dog

Now the second sentence doesn't feel right at all. Lets jazz it up a bit with punctuation.

Ran the dog?

Now we can see the old British King again and he seems to be asking if the dog ran, and specifically he wants to know if the dog *really ran*, as opposed to just, say, *walked*. To be even more specific, it feels almost like the King is asking a rhetorical question, but that may be going a bit too far.

It seems that in a natural language, the commutative rules don't always apply. More accurately, they seem to apply weakly in some cases and strongly in others. On a scale from one to ten, where ten is a perfect sentence, I would put

Am I?

at ten, however I would put the following sentence

Ran the dog?

at about 2, and that is only if the British king is shouting it.

In order to account for this, we need to improve our model and move away from the sleek and perfect group structures we saw in 4.1.

Rather than saying that NP and VP are commutative, maybe we can say they weakly commute. Remember that the computer model suggested so far will parse the sentence and provide a yes/no answer to whether or not the sentence is grammatically correct. To get this across mathematically, we assign a probability to the relation $NPVP \rightarrow S$. Now, given that NPVP is indeed a sentence, we can say that there is only a p percent probability that VPNP is a sentence. This is also a bit shy of the real situation since the degree to which the sentence is a good example, depends on the exact words. Recall how far off the sentence *Ranthedog* felt.

8. Representations Within The Superoperators

Given all the tools provided thus far, how can we mathematically get this sense of weak commutativity across? Let's look back at our group.

$$G = \{NP, VP, V, PP | I = NPVP, VP = VNPPP\}$$

Recall that not all sentences built in the form VP NP make what one might call, nice sentences. Though the sentence *am I* is very much a sentence, the sentence *Ran the dog* sounds a bit off. In a sweeping way, we can get this across by introducing a probabilistic grammar where $NPVP$ can be given a probability of 98 percent but $VPNP$ is down at 2 percent. Rewriting this in the form of a finitely presented group looks a bit odd.

(8.1)

$$G = \{NP, VP, V, PP | (I = NPVP, 0.98), (I = VPNP, 0.98), VP = VNPPP\}$$

Firstly, we no longer have a finitely presented group. In fact, we probably do not have a group at all. However, to say that what I have written above cannot be taken as a presentation of *some* algebraic structure is a bit harsh.

Assuming for the moment that this *is* a presentation for some algebraic structure, how would we find representations of this algebraic structure? Since this structure looks like it is a better model of natural language, it would probably be useful if we could find some representation. Since this paper concerns quantum computational linguistic models, I am going to suggest finding representations of this kind of structure within the operators on a Hilbert space.

Recall that we found representations of simplistic grammars in SU(n), by identifying the generators with particular elements of SU(n) which are specially chosen to exhibit the necessary algebraic properties. To find represenations of 8.1 we do the same thing, except we use an operator sum notation wherein we identify the generators with operators probabilities with the ...

John Hale, A Probabilistic Earley Parser as a Psycholinguistic ModelIn Proceedings of the Second Meeting of the North American Chapter of the Asssociation for Computational Linguistics. http://acl.ldc.upenn.edu/N/N01/N01-1021.pdf