Abstract

The development of quantum mechanics has led to some radical changes in the theory of computation. A quantum theory of computing has come up and has been applied to give fascinating theoretical results for problems such as prime factorization (Shor, 1994) for which polynomial-time algorithms are not yet known on a classical computer. In this paper, we take a look at whether they can speed up the existing algorithms for common tasks in Natural Language Processing. We look at the classical solution to the POS tagging problem via the use of the Viterbi algorithm and give a quantum version of the Viterbi algorithm which is faster than its classical counterpart.

1 Introduction

The main focus of this work is the development of quantum computing algorithms for problems in computational linguistics. This is mainly a theoretical study into how quantum computers can speed up tasks in NLP and we give a quantum algorithm for the famous Viterbi algorithm which is faster than the classical one. Part-of-Speech (POS) Tagging is the process of assigning grammatical (syntactic and morphological) categories (e.g. noun, verb, adjective, person, verbal class, gender etc.) to naturally occurring text. Annotated natural language texts are a useful preprocessing step for many natural language processing applications like parsing, information retrieval, and machine translation. The Viterbi algorithm (Viterbi, 1967) is a well-known dynamic programming scheme for POS tagging that treats sentence generation from a hidden tag sequence as a first-order Markov process and runs in time $O(T^2 L)$ where $T$ is the size of tag-set and $L$ is the length of sentence. At each stage of the Viterbi trellis, for each tag, we perform a search among the $T$ paths leading up to there and select one with highest score. This is an $O(T)$ task, which can be performed in time $O(T^{1/2})$ on a quantum computer. First, we present the basics of quantum computing and the reason for the improvements in time efficiency it gives.

2 Quantum Computing

A qubit or quantum bit is a unit of quantum information - the quantum analogue of the classical bit. A qubit is a two-state quantum-mechanical system, such as the polarization of a single photon. Whereas a bit must be either 0 or 1, a qubit can be 0, 1, or a superposition of both. The two states in which a qubit may be measured are known as basis states (or basis vectors). As is the tradition with any sort of quantum states, Dirac, or bra-ket notation, is used to represent them. This means that the two computational basis states are conventionally written as $|0\rangle$ and $|1\rangle$ (pronounced “ket 0” and “ket 1”). A pure qubit state is a linear quantum superposition of the basis states. This means that the qubit can be represented as a linear combination:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha$ and $\beta$ are probability amplitudes and can in general both be complex numbers.

An important distinguishing feature between a qubit and a classical bit is that multiple qubits can exhibit quantum entanglement. Entanglement is a non-local property that allows a set of qubits to express higher correlation than is possible in classical systems. Take, for example, two entangled qubits in the Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Imagine that these two entangled qubits are separated, with one each given to Alice and Bob. Alice makes a measurement of her qubit, obtaining $|0\rangle$ or $|1\rangle$. Because of the qubits’ entanglement, Bob must now get exactly the same measurement as Alice;
i.e., if she measures a $|0\rangle$, Bob must measure the same, as $|00\rangle$ is the only state where Alice’s qubit is a $|0\rangle$.

A number of entangled qubits taken together is a qubit register. Quantum computers perform calculations by manipulating qubits within a register. Taken together, quantum superposition and entanglement create an enormously enhanced computing power. While a 2-bit register in an ordinary computer can store only one of four binary configurations (00, 01, 10, or 11) at any given time, a 2-qubit register in a quantum computer can store all four configurations simultaneously, because each qubit represents two values. If more qubits are added, the increased capacity is expanded exponentially.

**Measurement** - Quantum Mechanics only provides probabilities for the different possible outcomes in an experiment - it provides no mechanism by which the actual, finally observed result, comes about. For an ideal measurement in quantum mechanics, also called a von Neumann measurement, the only possible measurement outcomes are equal to the eigenvalues (say $k$) of the operator representing the observable. Consider a system prepared in state $|\psi\rangle$. Since the eigenstates of the observable $\hat{O}$ form a complete basis called eigenbasis, the state vector $|\psi\rangle$ can be written in terms of the eigenstates as

$$|\psi\rangle = c_1|1\rangle + c_2|2\rangle + c_3|3\rangle + \cdots$$

where $c_1, c_2, \ldots$ are complex numbers in general. The eigenvalues $O_1, O_2, O_3, \ldots$ are all possible values of the measurement. The corresponding probabilities are given by

$$\Pr(O_n) = \frac{\langle n|\psi\rangle^2}{\langle \psi|\psi\rangle} = \frac{|c_n|^2}{\sum_k |c_k|^2}.$$ 

Usually $|\psi\rangle$ is assumed to be normalized, i.e. $\langle \psi|\psi\rangle = 1$. Therefore, the expression above is reduced to

$$\Pr(O_n) = |\langle n|\psi\rangle|^2 = |c_n|^2.$$ 

A quantum computer operates by setting the $n$ qubits in a controlled initial state that represents the problem at hand and by manipulating those qubits with a fixed sequence of quantum logic gates. The sequence of gates to be applied is called a quantum algorithm. The calculation ends with measurement of all the states, collapsing each qubit into one of the two pure states, so the outcome can be at most $n$ classical bits of information.

For example, if we prepare a 2-qubit system in the state $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{3}}|01\rangle + \frac{1}{\sqrt{6}}|11\rangle$, then a measurement on the system will yield results corresponding to the state $|00\rangle$ with probability $\frac{1}{2}$, state $|01\rangle$ with probability $\frac{1}{3}$ and state $|11\rangle$ with probability $\frac{1}{6}$.

### 3 The Quantum Minimum Algorithm

Let $T[0..N-1]$ be an unsorted table of $N$ items, each holding a value from an ordered set. The minimum searching problem is to find the index $y$ such that $T[y]$ is minimum. This clearly requires a linear number of probes on a classical probabilistic Turing machine. (Durr and Hoyer, 1996) gave a simple quantum algorithm which solves the problem using $O(N^{1/2})$ probes. The algorithm is as follows:

1. Choose threshold index $0 \leq y \leq N-1$ uniformly at random.
2. Repeat the following and interrupt it when the total running time is more than $22.5\sqrt{N} + 1.4\log^2 N$. Then go to 2c.
   (a) Initialize the register as a uniform superposition over the $N$ states, i.e., give each state a coefficient of $\frac{1}{\sqrt{N}}$. Mark every item $j$ for which $T[j] < T[y]$. This would be an $O(N)$ operation on a classical computer but here, the entire state which is a superposition of the $N$ basis states, is acted upon at once by a quantum operator.
   (b) Apply the quantum exponential searching algorithm of (Boyer, Brassard, Hoyer and Tapp, 1998).
   (c) Observe the register: let $y'$ be the outcome. If $T[y'] < T[y]$, then set threshold index $y$ to $y'$.
3. Return $y$.

### 3.1 Quantum Exponential Searching

This search algorithm receives as input a superposition of $N$ states, of which say $t$ are marked. It gives as output one of those $t$ states. The steps are:

1. Initialize $m = 1$ and $\lambda = \frac{6}{5}$.
2. Choose $j$ uniformly at random from the whole numbers smaller than $m$. 

3. Apply \( j \) iterations of Grover’s algorithm starting from initial state \( |\psi_0\rangle = \sum_i \frac{1}{\sqrt{N}} |i\rangle \).

4. Observe the register; Let \( i \) be the outcome. If \( T[i] = x \), then the problem is solved; exit and return \( i \).

5. Otherwise, set \( m \) to \( \min(\lambda m, \sqrt{N}) \) and go back to step 2.

3.2 The Grover Iteration

(Grover, 1996) shows that the unitary transform \( G \), defined below, efficiently implements what we called an iteration above. \( S_0 \) is an operator that inverts the sign of the coefficient of the \( |0\rangle \) state. Similarly, \( S_t \) inverts the sign of coefficients of all marked states. \( T \) is defined by its actions on the states \( |0\rangle, |1\rangle, \ldots, |N-1\rangle \) as

\[
T|j\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} (-1)^{(i,j)} |i\rangle
\]

where \( i, j \) denotes the bitwise dot product of the two binary strings denoting \( i \) and \( j \). Then the transform \( G \) is given by

\[
G = T S_0 T S_t
\]

Grover considers only the case when \( N \) is a power of 2 since the transform \( T \) is well-defined only in this case.

4 Viterbi for POS-tagging

4.1 The Classical Version

Firstly, given a tagged corpus and a tag-set \( t_1, t_2, \ldots, t_T \) of size \( T \), we learn tag-wise probabilities of starting the sentence \( \pi_t \); tag-to-tag transition probabilities \( P(t_j|t_i) \); and tag-to-word generation probabilities \( P(w_k|t_i) \). Then, given a sentence of length \( w_1, w_2, \ldots, w_L \) of length \( L \),

1. Initialize a 2-dimensional vector \( V \) of size \( T \times (L+1) \) to all zeroes.

2. Initialize a 2-dimensional vector \( B \) of size \( T \times L \) to null.

3. Set \( V[i][0] \) to \( \pi_t \) for all \( i \) in \( \{1, \ldots, T\} \).

4. For \( k \) in \( 1 \) to \( L \),

\[
B[j][k] = \arg \max_i P(t_j|t_i) \times P(w_k|t_i) \times V[i][k-1]
\]

\[
m = B[j][k]
\]

\[
V[j][k] = P(t_j|t_m)P(w_k|t_m)V[m][k-1]
\]

5. In the BNC corpus used by us, all sentences end in punctuations. So, we assign the corresponding tag to the last word in the sentence. Say, the index for this tag is \( p \). Then, \( tag_L = p \).

6. Now, we use the back-pointers stored in \( B \) to find the path that gave us maximum score and ended in \( p \).

For \( k \) in \( L-1 \) to \( 1 \), \( tag_k = B[tag_{k+1}][k+1] \)

4.2 The Quantum Approach

Note that step 4 in 4.1 has an inner loop that runs \( T \) times and needs to find the maximum among \( T \) quantities each time, leading to the \( O(T^2) \) component in the running time on a classical computer. On a quantum computer, the \( T \) quantities to be compared can be prepared together into a superposition of states in \( \log T \) time (because we need \( \log T \) qubits to represent \( T \) states) and then, we modify the Quantum Minimum Algorithm to get a Quantum Maximum Algorithm by changing the \(<\) comparison operator to \( \geq \) in \( 2a \). Since the \( T \) quantities for a fixed \( j \) and \( k \) are in a superposed state, we use an operator that fetches the required values from the \( V \) table and the probability values learnt during the training phase, simultaneously for all possible \( i \) in constant time. Then, the triplets are multiplied together to give the \( T \) quantities, again in constant time. Now, the Quantum Maximum Algorithm takes \( O(T^{1/2}) \) time to find the maximum among the \( T \) number of values, hence giving the reduction in overall running time from \( O(T^2 L) \) to \( O(T^{3/2} L) \).

5 Implementation and Results

We implemented a simulation of the quantum algorithm on a classical computer and assigned part-of-speech tags to the British National Corpus, which has a 61-strong tagset available at http://www.natcorp.ox.ac.uk/docs/c5spec.html. We padded this with 3 dummy tags in order to work with 64 states, i.e., 6-qubit quantum registers.

Additionally, instead of running the Quantum Minimum Algorithm for a time of \( 22.5\sqrt{N} + 1.4\log^2 N \), we restricted the number of iteration of step 2 to 10, thus giving a total running time of the same order (because the Quantum Exponential Searching over \( N \) states is an \( O(\sqrt{N}) \) operation). The rationale behind this was that the probability of getting the correct result from Grover search (of
which the Quantum Exponential Search is a generalization) algorithm oscillates with the number of iterations, rising quickly and peaking periodically. Thus, by performing a slightly lesser number of iterations, we do not lose out much on precision but save on execution time. (Note that the simulation of a quantum algorithm on a classical Turing machine incurs an exponential blow-up in execution time.)

We used smoothing in our implementation where the tag-to-word probability is boosted by $10^{-8}$ for all words. This ensures that even for words not present in the training corpus, a positive probability value is assigned. Pushing up the probabilities of solely the words absent from the training corpus can end up changing the tags of other words. To avoid this, the tag-to-word probabilities of all the words are increased by the same amount regardless of their original value. If there were no smoothing, the algorithm would end up leaving some words untagged. Due to smoothing no word is left untagged and hence the overall precision and recall values become the same.

Although the quantum Viterbi algorithm is probabilistic, the probability of success can be made high by setting the running time of the algorithm appropriately. Hence, the precision of the algorithm can be brought as close to that of the classical version as needed. For our implementation the classical version yielded a precision and recall of 0.9289 whereas the quantum counterpart yielded 0.9067 as the precision and recall.

### 5.1 Tag-wise Precision Analysis

The quantum Viterbi is a probabilistic algorithm and does badly in the case of tags with specific word forms. For example, VDI is the infinitive form of the verb DO, i.e. ‘do’ whereas VDZ is the -s form of verb DO, i.e. does. These are word forms with specific tags and the classical algorithm yields high accuracy in these cases. Here, the introduction of randomization by the quantum algorithm ends up lowering the accuracy significantly. For the tags VDI, VHN, VDB, VHI, VVB, VVI and EX0, the quantum implementation yielded a precision which was atleast 6% lesser than that given by the classical algorithm.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDI</td>
<td>0.9267</td>
<td>0.6285</td>
</tr>
<tr>
<td>ITJ</td>
<td>0.6152</td>
<td>0.3720</td>
</tr>
<tr>
<td>VHN</td>
<td>0.8174</td>
<td>0.6167</td>
</tr>
<tr>
<td>VDB</td>
<td>0.9754</td>
<td>0.8522</td>
</tr>
<tr>
<td>AJS</td>
<td>0.8782</td>
<td>0.7733</td>
</tr>
<tr>
<td>VDZ</td>
<td>0.9971</td>
<td>0.8968</td>
</tr>
</tbody>
</table>

Table 1: Tags for which Quantum Viterbi yielded higher precision.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDI</td>
<td>0.9642</td>
<td>0.9853</td>
</tr>
<tr>
<td>AJS</td>
<td>0.9553</td>
<td>0.9692</td>
</tr>
<tr>
<td>XX0</td>
<td>0.9584</td>
<td>0.968</td>
</tr>
<tr>
<td>VDZ</td>
<td>0.9709</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 2: Tags for which Quantum Viterbi yielded significantly lower precision.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHN</td>
<td>0.7679</td>
<td>0.4402</td>
</tr>
<tr>
<td>VDB</td>
<td>0.966</td>
<td>0.6866</td>
</tr>
<tr>
<td>VVB</td>
<td>0.7366</td>
<td>0.5359</td>
</tr>
<tr>
<td>AVP</td>
<td>0.8099</td>
<td>0.6563</td>
</tr>
<tr>
<td>ITJ</td>
<td>0.7809</td>
<td>0.65</td>
</tr>
<tr>
<td>AV0</td>
<td>0.9014</td>
<td>0.7929</td>
</tr>
<tr>
<td>CJIS</td>
<td>0.8030</td>
<td>0.7466</td>
</tr>
<tr>
<td>PNP</td>
<td>0.9537</td>
<td>0.9126</td>
</tr>
</tbody>
</table>

Table 3: Tags for which Quantum Viterbi yielded higher recall.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Classical</th>
<th>Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHN</td>
<td>0.7679</td>
<td>0.4402</td>
</tr>
<tr>
<td>VDB</td>
<td>0.966</td>
<td>0.6866</td>
</tr>
<tr>
<td>VVB</td>
<td>0.7366</td>
<td>0.5359</td>
</tr>
<tr>
<td>AVP</td>
<td>0.8099</td>
<td>0.6563</td>
</tr>
<tr>
<td>ITJ</td>
<td>0.7809</td>
<td>0.65</td>
</tr>
<tr>
<td>AV0</td>
<td>0.9014</td>
<td>0.7929</td>
</tr>
<tr>
<td>CJIS</td>
<td>0.8030</td>
<td>0.7466</td>
</tr>
<tr>
<td>PNP</td>
<td>0.9537</td>
<td>0.9126</td>
</tr>
</tbody>
</table>

Table 4: Tags for which Quantum Viterbi yielded significantly lower recall.

It is worth noticing that AV0 and AVP, both adverb forms suffered losses of $10 – 15\%$ in recall on using the quantum Viterbi, which suggests that these the probabilistic quantum algorithm for maximum-finding doesn’t do as well as the deterministic classical version on scores for words tagged AV0 or AVP. One possible reason is that the values among which maximum is to be determined are close to each other, in which
case a deterministic algorithm will go through but a probabilistic one has higher chances of failing. This is confirmed by the precision values being on the lower side (0.73 and 0.85) for these tags in the classical algorithm.

6 Conclusion and Future Work

In this paper, we have presented an algorithm for POS tagging problem in an alternate computing paradigm. Our algorithm is the quantum analogue of the classical Viterbi and brings down the time complexity of the same. The main contribution of this paper is theoretical, akin to all the major quantum computing algorithms.

Results obtained via simulation show that the decrease in precision due to the probabilistic nature of the algorithm is not very large and can be reduced by increasing the number of iterations. Hence, we are not trading off precision for time.

An application where the algorithm presented in this paper would result in a more significant reduction in time complexity is the machine translation of close languages. Among pairs of close languages such as Hindi and Urdu which almost follow word-for-word translation, we can treat the words of one language as part-of-speech tags for the corresponding words of the other and then, simply employ the Viterbi algorithm to obtain machine translation. Note that here, the number of states involved in the Viterbi trellis, i.e., $T$ would be of the order of $10^4$, and hence an $O(T^{1.5}L)$ algorithm will be much more efficient than an $O(T^2L)$ algorithm.

Also, POS tagging is just one problem. The design of quantum algorithms for other AI and NLP problems would be a widely open area or research.

Acknowledgments

We acknowledge the contribution of Dr. Pranab Sen from Tata Institute of Fundamental Research (TIFR), Mumbai, India in helping us during the design of the quantum Viterbi algorithm.

References


