A Goal-Oriented Language Model

Jason L. Hutchens, Michael D. Alder

***Abstract*— A "goal-oriented" language model is introduced. This model, or, more properly, family of traditional n^{th}–order markov models, is constrained by the value of a symbol which is to occur some time in the future. This constraint makes it possible to use the model to generate strings which have a goal in mind, a process which is difficult to achieve using standard markov models. An intended application of the model is as the language generation device of a non-intelligent conversation simulator, which is being developed for the 1999 Loebner contest in artificial intelligence. Another possible application is adaptive text compression.**

Keywords— **Turing test, Loebner contest, Markov models, text generation, data compression.**

I. Introduction

MARKOV models may be used generatively to produce mildly humorous gibberish, as demonstrated by Shannon in his classic paper [1]. Two 2^{nd}–order markov models are used in *MegaHAL*, a "non-intelligent conversation simulator", which was placed second in the 1998 Loebner contest in artificial intelligence, which is a version of the Turing test.

A requirement of the language model used in MegaHAL is that sentences should be generated from one or more keywords, so that they may bear a passing contextual relevance to the topic at hand. This is notoriously difficult to do with a traditional markov model, so an *ad hoc* solution was devised. A better solution was sought for the 1999 Loebner contest. After several false starts, a goal-oriented language model was developed. Apart from generating more convincing gibberish, which satisfies long-distance dependencies, this model may have applications in other fields, such as data compression.

In this paper, the process by which the model was developed is described, it is define rigorously, and some amusing examples of its use generatively are given. The paper concludes with speculation as to how the model could be applied to other fields, such as that of adaptive text compression. Although there has been some work in the past on long-range trigram models [2], it is believed that the present work is original.

II. The Loebner Contest

Alan Turing was a brilliant British mathematician who played a great role in the development of the digital computer. Turing's imitation game, nowadays known as *the Turing test*, was devised as a procedure for deciding whether or not a computer program is intelligent. The test takes place between a judge and two subjects; a human being and a computer program. Communication between the judge and the subjects occurs via a computer terminal, and it is the judge's task to decide which of the subjects is the human being. The human being attempts to help the judge to make the correct identification, while it is the computer program's task to trick the judge into making a false identification. If the latter case occurs, the computer program is said to be exhibiting intelligence [3].

The inaugural Loebner contest, held in 1991, was advertised as being the first formal instantiation of the Turing test, although it did not satisfy all of Turing's original specifications [4]. Since its inception, the contest has primarily attracted hobbyist entries which simulate conversation using ELIZA-style template matching, as developed by Joseph Weizenbaum at MIT in the mid-1960's [5].

The offer of a $2000 prize for the "most human-like computer program" prompted the primary author to submit his own ELIZA clone, named *HeX*, to the 1996 contest. This entry was successful, in that, although it didn't pass the Turing test proper (as is to be expected), it was judged the most human-like. This early success prompted a second entry in 1997, named *SEPO*, and a third in 1998, named MegaHAL. Both of these programs were placed second in a field of six.

III. Introducing MegaHAL

MegaHAL marked a departure from the standard conversation simulator entered into the Loebner contest [6]. A 2^{nd}–order markov model was trained on a small corpus of encyclopaedic information, non sequitir, popular quotations, film and television dialogue, and hand-crafted sentences. Such a model can be used to generate random gibberish. The question was how to use the model to formulate a suitable reply to the judge's input.

$$I(w_i|\mathcal{M}, s_{i-1}) = -\log_2 P(w_i|\mathcal{M}, s_{i-1}) \qquad (1)$$

The solution was simply to use the model to measure the information content of each word in the input sentence, as in equation 1, where $\mathcal{M}$ is the markov model, w_i is a random variable representing the i^{th} word in the judge's input, and $s_{i-1} = w_1, w_2, \ldots, w_{i-1}$ is the sequence of the $i-1$ preceding words. The 2^{nd}–order markov model approximates $P(w_i|\mathcal{M}, s_{i-1})$ as in equation 2, where $C(x)$ is a counting function which returns the number of occurrences of x in the training data.

$$P(w_i|\mathcal{M}, s_{i-1}) \approx \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \qquad (2)$$

J. Hutchens is with the Centre for Intelligent Information Processing Systems (CIIPS) within the Department of Electrical & Electronic Engineering at the University of Western Australia. E-mail: hutch@ciips.ee.uwa.edu.au

M. Alder is with the Department of Mathematics at the University of Western Australia. E-mail: mike@maths.uwa.edu.au

The information, $I(w_i|\mathcal{M}, s_{i-1})$, may be informally regarded as representing the "surprise" the word w causes the model $\mathcal{M}$. A keyword list, of the most surprising words in the input sentence, undergoes a heuristic transformation before being used to seed the generation process.[1]

Using a markov model to generate a random sentence containing a word specified *a priori* is difficult, because the model has no goal in mind—generation may be likened to a random walk. The MegaHAL program simplifies generation by using two markov models. The first is used to generate a sentence beginning from the keyword and ending at the end of the sentence, while the second model generates the sentence *backwards*, from the selected keyword to the beginning of the sentence.

$$\overline{I}(\mathcal{M}, s_n) = -\frac{1}{n}\sum_{i=1}^{n} I(w_i|\mathcal{M}, s_{i-1}) \tag{3}$$

Thousands of candidate replies are generated in this way, and the one with the highest average information, as in equation 3, is selected for display to the judge, under the assumption that generations with high average information tend to be both original (in the sense that they weren't encountered during training) and amusing (in the sense that they violate our expectations). It was thought that this was in the best interest of the program.

MegaHAL was placed second in the 1998 Loebner contest, ahead of four other more traditional conversation simulators. This, along with a newly introduced $25,000 prize,[2] encouraged us enough to put more thought into our 1999 Loebner contest entry.

IV. A New Approach

NonI, a non-intelligent conversation simulator destined for entry into the 1999 Loebner contest, has been designed to *UpWrite* the judge's input sentence to a more abstract level, conceivably representing the "concept" of the sentence. The UpWrite is a bootstrapping process used in syntactic pattern recognition [7] and would, in this case, abstract the judge's input by representing it as a short sequence of symbols. One possible UpWrite technique would be to select the word pair in the sentence which exhibits the highest mutual information, as in equation 4.[3]

$$I(w_x; w_z|\mathcal{M}) = \log_2 \frac{P(w_x, w_z|\mathcal{M})}{P(w_x|\mathcal{M})P(w_z|\mathcal{M})} \tag{4}$$

Having conceivable extracted the "concept" inherent in the judge's input in this way, a model would be used to formulate an appropriate response, which would be given in the form of a "concept". Finally, this response "concept" would be *DownWritten* to a sentence before being displayed to the judge as a reply. This is similar to what MegaHAL does—the UpWrite is the process which selects the word in the sentence with the highest information content, the reply "concept" is a heuristic transformation of this word, and the DownWrite is the sentence generated from the resulting keyword.

The DownWrite process requires a generative model. It would be preferable to use a stochastic language model merely because it is trivial to train such a model from a collection of example sentences, and that experience has shown that such models occasionally generate sentences which are surprisingly lucid and original. Capitalising upon the judge's tendency to anthropomorphise the program's responses is in our best interests.

We may formulate the generation problem as follows; the model is presented with a string of two or more symbols, and it is required to smooth the string in order to produce a sentence which is valid with respect to the model. This smoothing process is simply a matter of "filling in the blanks" between symbols until a valid sentence is produced. Consider, for instance, the string "▷`cat mat`◁", where the two control symbols, ▷ and ◁, delimit the boundaries of the sentence. One possible DownWrite of this "concept" string would be "▷`The cat sat on the mat`◁".

Perhaps the most obvious stochastic model for this purpose is one in which a pair of non-adjacent words provides a context for a probability distribution over the words which can occur anywhere between them. The distribution for the context $\{\triangleright, \triangleleft\}$ would be identical to the standard unigram distribution in this case. This model has the advantages that calculating $I(w_i; w_j)$ would be easy, and that generation would be recursive in nature; it would proceed much in the same way as a generation of a two-dimensional fractal "map" from control points (the inclusion of the buzzword "fractal" would be yet another arrow in our quiver, insofar as confusing the Loebner judges goes). The generation algorithm, `Generate(`w_x`,`w_z`)`, would proceed as follows:

if $\{w_x, w_z\}$ is not a valid word-pair **then**
  $w_y \Leftarrow$ random symbol inbetween $\{w_x, w_z\}$
  `Generate(`w_x`,`w_y`)`
  `Output(`w_y`)`
  `Generate(`w_y`,`w_z`)`
end if

This algorithm generates, and outputs, a sequence of symbols which "fill in the blanks" between the two specified as arguments. The symbol w_y is chosen at random according to the probability distribution of words which may occur anywhere within the context $\{w_x, w_z\}$; this distribution would be inferred in a Maximum Likelihood fashion from the data, as in equation 5. No symbol is generated if the $\{w_x, w_z\}$ is a valid word-pair; that is, w_x and w_z were adjacent in the training sentences, since, in this case, there are no blanks to be filled in between the words.

$$P(w_y|\{w_x, w_z\}) \approx \frac{C(w_x, \ldots, w_y, \ldots, w_z)}{C(w_x, \ldots, w_z)} \tag{5}$$

Calling `Generate(`w_x`,`w_z`)` with $w_x = \triangleright$ and $w_z = \triangleleft$ is sufficient to generate a random sentence. In order to generate a sentence according to the template $s_n = w_1, w_2, \ldots, w_n$, we would use the algorithm shown below:

```
Generate(▷, w_1)
for i = 1 to n − 1 do
  Output(w_i)
  Generate(w_i, w_{i+1})
end for
Output(w_n)
Generate(w_n, ◁)
```

In all cases, the `Output`(w_x) function takes whatever steps are necessary to store the generated sentence; it may, for example, simply print w_x to the console. We give some examples of random generations below for a model which was inferred from 174 sentences taken from the Probert E-Text Encyclopaedia, Edition 10.0 [8]:

```
A sword is an offensive weapon designed
chiefly for the sale and consumption of deep
unconsciousness.

Dance is a person who suffers from the carcass
of movement.

Cement is information, especially that stored
in a small, usually tree dwelling primate.
```

These three example sentences were selected for their amusement value. Note that the sentences do exhibit long-distance dependencies, as in the second sentence, in which the words `Dance` and `movement` are related. Also, note that the stopping criteria used, that word-pairs must be adjacent, guarantees that the sentences are grammatical within a window of about three or four words, provided that the training sentences are grammatical. This is the same level of localised grammaticality which we would expect from a 1^{st}–order markov model.

It was found that, although the model did generate sentences which we would classify as adequate, a large proportion of generations failed to halt within reasonable time. This is due to the rather *naïvé* stopping criterion; that the symbols need to be a valid word pair—that is, that they need to have been observed as adjacent symbols during training. All too often the most common word pair {`the`, `the`}, which is not valid, would generate the most common word `the`, resulting in recursion which might as well be infinite.

All of the heuristic measures introduced to counteract this problem caused problems of their own; usually they would result in generations which failed due to the fact that a word-pair in the generated sentence did not exist in the model. Rather than introduce more complicated stopping criteria, it was decided to further constrain the probability distributions in the model.

This was achieved by updating our estimate of the probability distribution using only the words which occur immediately after the left-most word in the word-pair, as in equation 6. This constraint sufficiently restricted the model, so that all generated sentences were satisfactory (in length, if not in content).

$$P(w_y|\{w_x, w_z\}) \approx \frac{C(w_x, w_y, \ldots, w_z)}{C(w_x, \ldots, w_z)} \tag{6}$$

It soon became clear that this model was equivalent to a family of 1^{st}–order markov models, each of which is constrained by a a different symbol from the alphabet. We use M_w^n to denote an n^{th}–order markov model which is constrained by the symbol w, which occurs somewhere in the future. It is this symbol which is the goal of the language model. Note that $\mathcal{M}_{\triangleleft}^n$ represents the traditional, unconstrained, n^{th}–order markov model.

V. A Goal-Oriented Language Model

The goal-oriented language model will obviously require considerably more storage than a standard markov model of the same order. In fact, because the constraining future symbol may occur *anywhere* between the next symbol and the end of the sentence, it will require more storage space than a traditional markov model of order $n + 1$.

Inference of an n^{th}–order goal-oriented language model is a simple process. We define a string s_i to be a *prefix string* of a string s_n if the first i words in each strings match, and $n \geq i$. For each substring s_i of the training sentence s_n, we update the statistics of model $\mathcal{M}_{w_i}^n$ with the string s_{i-1}, which is a prefix of s_i, using the regular Maximum Likelihood inference process.

Generation using the goal-oriented language model, and a template sentence, is a straightforward process. We begin by verifying that it is possible to generate a sentence at all from the template string s_n. We do this by checking that each triple of words w_{i-1}, w_i, w_{i+1} in s_n is valid in the sense that the triple was also present in at least one of the training sentences. If it has, we can guarantee that we can generate a sentence from that triple, because we have already seen at least one sentence containing it (and that sentence, therefore, could be our generated one). If all triples pass this test, it means that generation of a sentence which contains all of them is possible, provided that we allow the language model to "fall back" when necessary, a process which is introduced below, following the discussion on generation.

Generation is a linear process as with standard markov models, as opposed to the appealing recursive process described previously. We generate from the template sentence $s_n = w_1, w_2, \ldots, w_n$, where $w_1 = \triangleright$ and $w_n = \triangleleft$, by using the model $\mathcal{M}_{w_2}^n$ to estimate a probability distribution over the possible sequent symbols of $w_1 = \triangleright$. If the target symbol, w_2, is assigned a non-zero probability according to this distribution, then we have reached our goal. Otherwise, a word w_x is selected at random, according to the estimated probability distribution, and the process is iterated. Once the target is reached, we simply stop using $\mathcal{M}_{w_2}^n$, and switch to $\mathcal{M}_{w_3}^n$ instead. The process continues until the target $w_n = \triangleleft$ is reached.

It may be possible, for an n^{th}–order goal-oriented language model, that the context of words generated thus far

does not exist in the model. This is due to the slight discontinuities in the generated sentence caused by the sudden switch from one constrained markov model to another. It is not worth our while to further constrain the generation so that this never occurs; such a process is computationally expensive (we would need to validate every triple in the generated sentence thus far, and, if this validation failed, we would need to back up to a point where it succeeded). Rather than constrain generation, we choose to "fall back" to a simpler model, of order $n-1$, until the model is able to make a prediction. In the worst case, we would fall back to the 1^{st}–order model. In practice, it is planned to use the 1^{st}–order model exclusively, which renders this a moot point.

To give a brief demonstration of this model in action, a 1^{st}–order goal-oriented language model was inferred from the same 174 sentences used in the previous example. The word-pair with the highest mutual information was found for each of these training sentences, and these word-pairs were used to constrain the generation of random sentences. A selection of hand-picked example generations are shown below:

```
An acronym is a curved wooden weapon of other
words.

A symbol is a suspended brass disk which
represents something else.

Bone is a hollow shell filled with the
external coating of an animal.

Leonardo da Vinci was an Italian artist and
expert in Kung Fu who popularised the martial
arts in unpublished note books.

The nose is an English naturalist.  He
published his theory of smell.
```

These example sentences exhibit long-distance dependencies as a result of the constraints imposed by the template used to generate them. Thousands of generations were performed, and all of them were successful, in that generation halted after a reasonable period of time, and the generated sentences had about the same average length as the training sentences.

VI. Potential Applications of the Model

Although using models generatively is fun, one may ask whether there are any serious applications of this work. We believe there are, and choose the application of data compression to illustrate our point.

Statistical data compression uses a predictive model to estimate the probability of the next symbol in a sequence, and arithmetic coding is typically used to encode the symbol for transmission. This paradigm seems to suggest that symbols need to be transmitted sequentially, but this isn't the case. The only requirement is that the receiver constructs the same predictive model as the transmitter. Therefore, the model can be based only upon the symbols transmitted thus far.

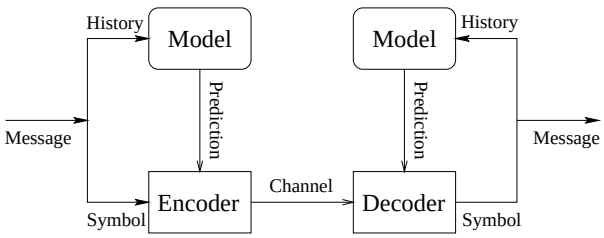

Fig. 1. An adaptive statistical compression system

It may be useful, for instance, to transmit symbols out-of-order, if the constraints imposed by doing this, with respect to a goal-oriented language model, sufficiently improve the performance of the compressor. In order to accomplish this, the receiver must be made aware of what the sequence of transmission is. Intuitively, one feels that this would require additional information to be sent; information which would negate the advantages of transmitting non-sequentially. Again, this need not be the case. Either the receiver and transmitter could agree upon an order of transmission, or such an order could be inferred from the statistics of the data itself.

For instance, the transmitter could send the n^{th} symbol, and then all of the symbols up to (but not including) this symbol sequentially. It could then send the $2n^{th}$ symbol, and so on. There may be no good reason for supposing that the n^{th} symbol restricts the statistics of the model enough to improve compression, but this method provides a way of making the transmitter and receiver agree on the sequence. And, intuitively, it seems that it is unlikely that the constraint imposed by the n^{th} symbol should be a handicap.

A second technique, of inferring the transmission order from the model, is as follows. The transmitter scans ahead n symbols, calculating the mutual information between the most recently transmitted symbol and each of the n symbols which occur in the future. Whenever the highest mutual information exceeds some pre-specified threshold value, the future symbol which caused this high value is transmitted instead of the next symbol, in the belief that this will constrain the probabilities of the symbols which occur between the two. In the case where the next symbol is the one with the highest mutual information, the transmitter will somehow have to make the receiver aware of this, perhaps by transmitting a special NULL symbol.

VII. Conclusion

In this paper we have introduced a novel language model; one that is constrained by a goal symbol. We have defined the model rigorously, and illustrated its use as a generator of gibberish. Apart from this amusing non-application, we have speculated on how the model may be used in other domains, concentrating our discussions on adaptive text compression. Future work includes the use of the model in NonI, a non-intelligent conversation simulator, implementation of a text compression scheme using the model, and further investigation of other potential applications.

Notes

[1] This heuristic transformation was added to increase the realism of the replies. Keywords such as "I" and "you" were transformed to "You" and "me" so that their appearance in the program's reply made more sense.

[2] This prize is on the brink of attainability. It is to be awarded to the program which is erroneously identified as a human being by the judges more than 50% of the time. Although such a task seems impossible, restrictions imposed by the Loebner contest, such as judges only getting twelve minutes to converse with each computer terminal, have prompted us to take things a bit more seriously than in previous years. Realistically, though, we still consider the prize to be unwinnable, and continue to treat the Loebner contest as an enjoyable annual diversion from serious research.

[3] In which case it would be rather cavalier of us to proclaim that the result encapsulates the "concept" of the sentence. We use the term "concept" in a rather loose way here; with the intention of referring to the fact that the "concept" may often have the same meaning to the listener as the complete sentence. More ambitious UpWrites may take world knowledge into account, along with the history of dialogue with the judge. We opt for the easiest solution merely because the more ambitious ones seem unattainable using current technology.

References

[1] Claude E. Shannon and Warren Weaver, *The Mathematical theory of Communication*, University of Illinois Press, 1949.

[2] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ureš, "Inference and estimation of a long-range trigram model," in *International Colloquim on Grammatical Inference*, 1994, pp. 57–65.

[3] A.M. Turing, "Computing machinery and intelligence," in *Collected works of A.M. Turing: Mechanical Intelligence*, D.C. Ince, Ed., chapter 5, pp. 133–160. Elsevier Science Publishers, 1992.

[4] Jason L. Hutchens, "How to pass the turing test by cheating," Available at: `http://ciips.ee.uwa.edu.au/Papers/`, 1996.

[5] Joseph Weizenbaum, *Computer Power and Human Reason*, W.H. Freeman and Company, 1976.

[6] Jason L. Hutchens, "Introducing MegaHAL," in *NeMLaP3 / CoNLL98 Workshop on Human-Computer Conversation, ACL*, David M. W. Powers, Ed., January 1998, pp. 271–274.

[7] Michael D. Alder, "Inference of syntax for point sets," in *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*, E. S. Gelsema and L. N. Kanal, Eds. June 1994, number 16 in Machine Intelligence and Pattern Recognition, pp. 45–58, Elsevier Science B.V.

[8] "The Probert e-text encyclopaedia," Available at: `http://www.pins.co.uk/upages/probertm/`