

Compression of parallel texts

Craig Nevill and Timothy Bell*
Department of Computer Science
University of Canterbury
Christchurch
New Zealand
phone (+64 3) 642 352
fax (+64 3) 642 999
e-mail tim@cosc.canterbury.ac.nz

* Address for correspondence: Timothy Bell, Department of Computer Science, University of Canterbury, Christchurch, New Zealand. The authors are grateful to Bruce McKenzie and Rodney Harries for their helpful comments, to Alistair Moffat for supplying the PPM program, to Greg Ewing who performed some of the programming, and to the subjects of the gambling experiments.

Abstract

The world-wide use of digital storage and communications devices is increasing the need to make texts available in multiple languages. To minimise the cost of storing and transmitting multiple translations of a text, one could store the text in just one language, from which other translations can be created. Unfortunately, the quality of machine translation techniques is not good enough for this to be feasible. An alternative is to store a compressed form of translated versions of a text, taking advantage of the availability of the original text. The original text provides some of the semantic content of the text that is to be compressed, and therefore makes it possible for compression to be more efficient than if that information were not available.

This paper reports investigations into the use of a parallel text to represent its translated version compactly. We begin with an experiment to evaluate the information content of a text when a parallel translation is available. This is achieved by having human subjects guess texts letter by letter, with and without a parallel translation. The perceived information content of a text can be determined from the way subjects make their guesses. The design and results of this experiment are described. The main conclusion is that while the text is considerably more predictable with the aid of a parallel translation, there is a surprising amount of information introduced by the translation.

Insights obtained from this experiment are then applied in the design of a mechanical system for compressing parallel texts. The system stores one translation of a text intact, and then compresses further translations of the text with the aid of the original. The method described is able to compress texts significantly better than is possible without the aid of a parallel text. Aspects of the design are also applicable to future compressors that might take advantage of the semantic content of a text to obtain better compression.

1 Introduction

Traditional methods of recording and disseminating textual information on paper are rapidly being replaced by the new medium of digital storage and communications. This facilitates the efficient and flexible retrieval and analysis of information. Improvements in global communications have brought into contact geographically isolated communities who wish to share information, but each would like to do this in their own language. The problem of translating documents has received considerable attention, but here we concentrate on the *dissemination* of the information to individuals so that they can study it in the language of their choice. The essence of the problem is the

efficient storage and transmission of the same information in a large number of translations. Although the storage capacities of digital devices are growing rapidly, the demand for storage is easily keeping up and so it is worthwhile to investigate methods for storing such data as compactly as possible. Then, with multiple-language versions of a text being stored and transmitted together efficiently, the choice of language for the presentation of the text can be deferred to as late as desired, and it is even possible to view two or more versions of a text simultaneously.

In general, parallel texts are natural-language texts that have the same semantic content, but are expressed in different forms. Specific examples of parallel texts in current use are official Canadian documents, which appear in both English and French; instructions for electronic appliances, where the product is sold in several countries; and versions of the Bible, which originate from the same source but are translated differently. Software houses now market their programs internationally, and systems are being devised to produce a language-independent product, where the messages that the program produces are easily adapted to the local language. Future possibilities include a multilingual encyclopædia or novel, where the user can select at any point the language in which they would like to view the information.

The disadvantage of such a flexible approach is the cost of storing and transmitting the multiple versions of a text. If fast, accurate machine translation were possible then the problem would be solved, since only one version of a text would be needed to produce other translations on the fly. However, this is not likely to be the case in the near future, and so here we explore efficient ways of storing a text *after* it has been translated, when it is to be stored with the original text.

From an information theoretic point of view, accurately translated copies of the original text would be expected to contain almost no extra information if the original text is available, so in principle it should be possible to store and transmit these texts with very little extra cost. This paper investigates methods for storing such parallel translations more efficiently by taking into account the close relationship between a translated document and its source (or two documents that have been translated from the same source). The aim of the work is to extend traditional methods of text compression to the parallel text situation. Compression is a type of coding that reduces the amount of space required to represent a text, where the original text can be reconstructed exactly from the coded version. There are two main parts to the present investigation: an experiment involving human subjects to evaluate the amount of extra information in a parallel text, and some practical techniques to store a parallel text compactly.

Section 2 briefly reviews the principles of text compression, and explains the relationship between parallel texts and compression. Sections 3 and 4 describes the design and results of a “gambling” experiment to determine the possible usefulness of a

parallel text for compression, and Section 5 details the results of an experiments in implementing parallel text compression.

2 Compression and prediction

In the 1940s, Shannon heralded the birth of a new discipline, *Information Theory*, with a paper showing how the predictability or redundancy of a text was related to its information content, or *entropy* [Shannon 1948]. His thesis was essentially this: if it is possible to predict with a high degree of accuracy what will follow a given portion of text, the amount of information contained in the text predicted is small. The entropy, or lack of predictability of the text is where the actual information lies. Compression involves storing only the information contained in the text by employing a different representation of the text. The redundant part of the text is effectively eliminated, as it is predictable. The original text is re-created exactly when the information is decompressed. In this section we introduce some relevant principles of compression. Fuller descriptions of the subject can be found in [Bell et. al. 1989, 1990; Lellewer and Hirschberg 1987].

Modern compression techniques rely on *predictions* about what will appear next in a text. Predictions are made by a *model*. A model does not usually make absolute predictions (e.g. the next word is “the”), but probabilistic predictions (e.g. the probability that the next word is “the” is 95%). Many different models of a text can be constructed, ranging from those that look only at the frequency of individual letters, through those that consider groups of letters, up to the most sophisticated models that exist only in the human mind. The latter are based on an understanding of the text, on experience with different texts of the same genre, and on other cognitive processes. The better the model of the text, the more redundancy is apparent, and the more concisely the information can be stored. An example of a poor model of text is one that considers every character to be equally likely (this corresponds to the type of coding used in systems such as ASCII and EBCDIC). The inexactness of a model does not prevent a text from being recreated exactly after compression; it simply means that the compressed form of the text will be less concise than if a better model had been used.

Computers achieve compression by forming relatively simple models of texts, usually based on a statistical analysis. These models estimate probabilities for symbols in the text, and the symbols are coded based on the probabilities. Likely symbols are given short codes, and unlikely symbols are given long codes, so that most of the time shorter codes will be used. The compression system has two parts, a compressor and decompressor, which are separated either spatially or temporally. Each has the same model of the text available; the compressor uses the model to encode the text, and the decompressor uses it to interpret the codes that it receives from the compressor. Considerable attention has been given to discovering better modelling techniques, as the

amount of compression is dictated by the model. The better the model of a piece of text, the better the predictions made by the model, the less redundancy is stored, the smaller the compressed text is, and the better the compression: thus prediction is tantamount to compression.

Let us examine more closely the relationship between probabilistic prediction and compression. A measure of the information content of the symbol s is the *entropy* $H(s)$. If s is predicted with probability $p(s)$ then its entropy is

$$H(s) = -\log_2 p(s) \text{ bits.}$$

The unit of measure is the *bit*, which is equivalent to one yes or no, true or false answer. This means that if you are 50% sure that the letter after “The quick brown fo” is an “x”, and that letter occurs, the information content is $H(\text{“x”}) = -\log_2 0.5 = 1$ bit. If however the phrase is “The quick brown fowl”, and an “w” was assigned a probability of 2%, the information content of the “w” would be $H(\text{“w”}) = -\log_2 0.02 \approx 5.6$ bits. Following this relationship, if a symbol is certain to occur ($p=1$) then it contains no information.

A fundamental result of information theory is Shannon’s source coding theorem [Shannon 1948], which essentially establishes that the entropy of a symbol is the optimal number of bits that should be used to code the symbol. In the examples above, an “x” should contribute 1 bit to the output of the compressor, while an “w” should contribute 5.6 bits. Of course, the entropy depends on the probability estimates, and more accurate estimates will achieve better compression.

It is common to measure compression by averaging this entropy over an entire text. Thus if a text of 100 characters has a total entropy of 200 bits then we express the average as 2 bits per character. The source coding theorem implies that on average the best representation of the text will use 200 bits. In practice the 100 characters were probably stored originally as 100 bytes, and so the file is now 25% of the original size.

Transforming the probabilistic predictions made by the model into a more concrete form is called *coding*. While modelling is somewhat of an art, with many refinements yet to be made, coding is a solved problem. The technique of arithmetic coding [Witten et. al. 1987] provides a method of representing a series of characters, using the probabilities assigned to them by a model, in almost exactly the number of bits dictated by the entropy. Even if the entropy of a symbol is non-integral (for example, 5.6 bits), arithmetic coding is able to achieve the entropy[†]. The output from an arithmetic coder can be stored, transmitted and eventually reconstructed with the aid of the same model.

[†] The more traditional method of Huffman coding [Huffman 1952] performs a similar function to arithmetic coding, but cannot work with non-integral code lengths. Consequently it codes less efficiently.

Since arithmetic coding is optimal for the probabilities given, we need to concentrate on the *models* of text, knowing that they can be combined with arithmetic coding to produce a practical compression technique.

One final note about models. Most good models of text adapt as they find out more about the text that they are modelling. Human models tend to quickly classify the text subconsciously, and use experience with that genre as a starting point for the model. After that point, both human and automated models adapt to the specific vocabulary and style of a text, getting better as they learn more. In the case of a computer model being used for compression, the compression usually improves as processing proceeds. This is known as *adaptive coding*. Although it might seem to lead to anomalies because the compressor is changing its model without informing the decompressor, the decompressor has available all of the text previously decoded, and so is able to adapt its model in a manner identical to the compressor. It has been shown that it is generally more efficient for a compressor and decompressor to construct their own models adaptively than for the compressor to pre-scan the entire text, construct a model, and send it to the decompressor [Cleary and Witten 1984a].

A parallel translation of a text offers considerable assistance in making predictions. For example, after the sequence “In the ...” the next symbol is very difficult to predict without more information. However, if we know that a parallel French translation of the text is “Au commencement Dieu créa le ciel et la terre” then the probability of the next word in the English text being “beginning” is very high, and it can be coded very efficiently.

The compression of parallel translations is simplified considerably when the texts are in the same language. A different-language pair of texts can be converted to a same-language situation by applying a mechanical transliteration to one of the texts. The decompressor performs the same mechanical translation as the compressor, and then both can use the (roughly) translated text to enhance predictions for the correctly translated text being compressed. The mechanical translation need not be of high quality—in fact, it could just be a word for word translation—but the more accurate it is, the better the compression that can be achieved. If the machine translation somehow produces the correctly translated text exactly then predictions will be very accurate, and the compressed form will be negligibly small. Thus, the availability of even crude mechanical translation means that same-language techniques generalise immediately to the different-language situation, we can concentrate on compression of same language pairs.

As an example, consider the compression system in Figure 1. A French text is available to the compressor and decompressor, and a parallel English version of the text is being compressed. The French text, the English text so far, a machine translation and thesaurus are available to both the encoder and the decoder. A machine translation of

the French text has been made at both ends. It is accurate in meaning, but translates “commencement” as “start” rather than “beginning”. The model, however, hedges its bets by using a thesaurus to generate several possibilities apart from “start”, and predicts “beginning” with a 25% probability. The encoder receives the word “beginning”, and as it has been predicted with 25% probability, it transmits the word in $-\log_2 0.25 = 2$ bits. The decoder receives the two bits, and using the predictions made by the identical model at the receiving end, produces the word “beginning”.

3 Design of a prediction experiment

In the last section it was noted that the best models of natural language texts currently exist within the human mind. The field of computational linguistics bears witness to the difficulty of developing a computational model of natural language, but it is one of the human race’s fortes. If a human is asked to make predictions about a text we can expect them to perform particularly well, and their performance indicates a bound for the compression performance that might be expected from a computer.

The design of an experiment to elicit probabilistic predictions from human subjects requires some care. We asked subjects to predict a text letter by letter, betting money on the possible letters, the bets being directly related to the perceived probabilities of characters. This method is an extension of work by Shannon (1951), Cover and King (1978), and Zunde and Kelman (1985).

Shannon originally tried to place bounds on the entropy of English by asking subjects to guess the next letter of a text until the correct letter was chosen. The number of guesses needed to select the correct character was analysed to determine bounds for the entropy of that character. Because the number of guesses does not indicate the probability exactly, the bounds are quite loose.

Cover and King achieved tighter bounds by requiring subjects not only to guess a letter, but also to say how sure they were that they were correct by having them bet on the outcome—introducing the *gambling* concept. In this method a subject is given some initial capital, and places a bet on what they think the most likely character. If they are wrong then they bet some of the remaining capital on another letter. A return is paid in proportion to the amount of capital bet on the correct character, and that return becomes the capital for betting on the next character. In the experiments subjects are encouraged to maximise their winnings. It can be shown that this minimises the entropy reflected by their bets. To maximise winnings, they must place their money according to their best judgement. In other words, they must be risk neutral — a risk taker will lose more because they will have less to bet on other letters when their initial guesses are wrong, while a risk averse person will not obtain the optimal return for their bet each time. Thus the subject is well motivated to make good probability estimates. The percentage of the money that the gambler bets on a letter should

therefore reflect the likelihood of that letter according to their internal “model” of the text.

Zunde and Kelman improved the experiment by introducing *composite betting*, where subjects can choose sets of characters to bet on, narrowing down their options until they find the correct one. Subjects may still bet on single characters if they wish, but composite bets make it faster to narrow down the correct character when none is particularly likely. Zunde and Kelman also automated the process using a computer to record bets.

In this way, the entropy of the text being predicted with respect to the gambler’s internal model can be determined. The lower bound on this entropy over several gamblers indicates the actual information content of that piece of text with respect to the best possible model.

To see how the gambling predictions relate to compression, imagine a pair of identical twins who have the peculiar property of thinking about a problem in exactly the same way. Given identical texts, they will assign exactly the same probabilities to the possible letters. If the predictions of one of the twins are used as the basis for a compressor, using arithmetic coding to transform the probabilities into bits, the text can be encoded using the exact number of bits dictated by the entropy of the probabilities assigned to the letters. The other twin’s predictions could then be used as the basis of a decompressor, enabling the original text to be reconstructed. This situation has its analog in automated compression, where the twins are two identical programs running on two identical machines. Machines, of course, are not as clever as humans, although they are more consistent!

The aim of the experiments in the context of parallel text compression was to ascertain the amount of help a parallel text is in the prediction of a given text. Two sets of gamblers were asked to predict a portion of text, given the preceding text for about ten pages. One group was also given a parallel text up to and past the point at which the corresponding unknown text started.

For example, subjects were given the text up to and including “What do you want? Tell me and you”, where the text that they had to predict character by character was “shall have it.” The parallel text available to some of the subjects was “What is thy petition, queen Esther? and it shall be granted thee:”.

The betting was automated, using a program written in the Apple Macintosh *Hypercard* environment. The layout of the screen is shown in Figure 2. Operation is by direct manipulation rather than by commands, which made the program simpler to use, and aided understanding of the concepts involved. The gambler uses a mouse to select a letter or group of letters from the menu on the screen, and chooses a probability using the slider at the right. The scale on the slider is marked with words describing the confidence that each level implies, and a corresponding percentage is indicated in a

box above the keyboard. Thus the subject can either specify a probability for the letter chosen, or can think of the bet in descriptive terms (“very likely” etc.). Information on the effects of the gamblers choice if they are right or wrong is displayed at the top left, so they can experiment with the effects of a bet before making a commitment. As letters are eliminated, they disappear from the menu, making the available choices clearer and avoiding errors.

It is unproductive for subjects to bet more than 95% of their capital on one symbol because the payoff beyond this is only marginally better, but the consequences of an incorrect bet are disastrous. In initial experiments, an over-confident subject would occasionally bet 99.9% on the wrong symbol. They were then left with almost no capital to bet on the correct symbol, which devastated their winnings. In terms of compression, 95% corresponds to 0.07 bits, while 99.9% corresponds to 0.001 bits. Although the latter is a lot smaller, both are negligible compared with the typical average of about 1 bit per symbol. Therefore, since it is in a subject’s best interest, the slider was calibrated so the maximum bet was 95%. Also, a percentage below the breakeven probability, the probability which would be assigned if all characters were equally likely, cannot be selected. If the gambler were to select such a percentage, it would imply that they believe the *complement* of the selected letters to be more likely; they should therefore select the complement and bet a higher amount.

Results were gleaned from log files which were kept by the betting program. The log files contained information about the letters that were selected, what probability was given, and the time of each bet. An example of a typical betting sequence for a letter is shown in Figure 3, a sample of the log file for one of the better gamblers. The subject has just predicted the characters “If it pleases your majesty to grant my h”. The following characters are “umble request”. The log reveals that the subject has difficulty predicting the first “u”. After a few minutes thought, he decides to find out if the letter is a vowel. This is successful, and so 80% of the capital is still available to establish which vowel it is. He is evidently surprised that it is a “u”, although the 7% probability is considerably more than the breakeven probability, which is 3.7%. After the “u” is predicted, predictions are more confident, with the next character being guessed correctly the first time, after only a minute. As the word unfolds the subject bets more of his capital and takes less time to decide, indicating a high level of confidence, and a low amount of information in the remaining characters.

It is important that subjects are well motivated to do well in the experiments. One way of achieving this is to pay them according to their performance. Since we were not in a position to gamble with the funds of our institution, a more satisfactory approach was to offer an attractive trophy as a prize for the best gambler. The whole experiment was run as a tournament between the subjects, with heats, semi-finals and a final. A tournament has several benefits for the experiment. Not only did the air of competition

encourage gamblers to do their best, but the *best* subjects worked on three texts, generating more data from the most adept gamblers. Using a tournament also meant that several texts could be used, since only subjects competing in the same heat had to use the same text. This is particularly important for parallel text compression, since results are required for subjects with and without parallel texts, yet a subject with a parallel text would otherwise have an unfair advantage over one without.

Conducting the gambling experiments is a surprisingly difficult and time-consuming task. It involves finding and timetabling subjects, teaching them how to use the betting program and explaining the concepts of prediction, betting and entropy. A practice run is essential, since a mistake in the actual experiment cannot be undone—one cannot allow a gambler to revise his choices after the outcome is known!

The experiment is also demanding on the subjects. The several hours of concentration required are quite exhausting, and a break is needed from time to time to maintain good performance. Cups of tea and coffee were received with enthusiasm.

4 Results of the human predictions

We conducted experiments with three texts, each with and without the aid of a parallel text. Table 1 shows the sources of the texts that were predicted, and the parallel text that was provided to some subjects to aid predictions. The final entropies as perceived by the subjects are given in the last two columns. Where several entropies are shown for a text, they are the results from several subjects. The entropy of unaided predictions of English text are consistent with prior results (Shannon 1951, Cover and King 1978), which suggested values of around 1.3 bits per character.

The entropy of 3.4 bits per character for text 1 without a parallel text was due to a subject who had trouble understanding the concepts involved and the operation of the betting program (the program was an earlier version of the one described in Section 3, and was modified to avoid the problems encountered). The 1.7 bits per character was the result of a similar situation. Nevertheless, some subjects will naturally do better than others, and since we require a lower bound for the entropy, it is reasonable to give considerable weight to the best results, bearing in mind that there is a small chance that a subject might happen to be lucky in one experiment.

Looking at the results of the better subjects, the typical entropy of those supplied with a parallel text was significantly lower than that those without: about half for the first text, and about one third better for the second text (although more samples would be needed to draw conclusions about this text). The third parallel text seems even harder to predict, with only one fifth improvement. This is not surprising, because the translation of the latter text was more liberal.

The results of the experiment have shown that parallel texts are helpful, but their usefulness varies. It is surprising how much extra information the translations contain. This is interesting because it gives an indication of how much information in a text relates to the essential *meaning* of the text, and how much to *style*. In giving a subject a parallel text, they are supposedly given the *meaning* of what they are about to predict. The only surprises that they will encounter are the way in which that information is *conveyed* — the vocabulary, grammar and syntax, and the difference in writing style of the two authors/translators. This information might initially be thought to be small compared to the actual meaning, but these experiments indicate that as much as half of the information contained in a text is stylistic, not semantic. One might expect the information content of a text when a parallel text is available to be negligible, but the mediocre improvement with a parallel text does not bear this out. This has interesting implications for those involved in machine modelling and comprehension of natural language — it may be necessary to place emphasis on the *expression* of meaning as much as the meaning itself.

5 An implementation of parallel compression

Throughout the design of a mechanical system to compress parallel texts, constant recourse has been made to the human model — how would a human predict text given a parallel text? The human model in this case can be considered to have two components — knowledge about English text quite apart from the parallel text, and the special clues provided by the parallel text. When nothing sensible seems to be suggested by the parallel text, a human will rely simply on the style and meaning of the target text, as well as on their general grasp of English expression. If the parallel text suggests very strongly what is about to appear (particularly when novel material is encountered) then it is used to make a prediction. This subconscious model-switching means that prediction need be no worse than what it would be without a parallel text. A parallel text may even be a hindrance, if it is not close enough to the target to be useful, but mostly it will be useful to some degree. In reality, a blend of the two models is used with each model contributing to the prediction in proportion to the confidence of its prediction.

Our goal was to use a parallel text to compress better than the best compression possible without a parallel text. It seems reasonable to implement the same model-blending techniques in an automated system that a human would use. The marriage of a model specific to the parallel text and a general-purpose model is achieved by having both models assess how confident they are about their predictions. A parallel model will sometimes be totally stumped, if the letters appearing in the target text do not seem to correspond to any word, or synonym of the source. In this case, the general model takes over. Sometimes the opposite may be true; the parallel model will take over when

the general model runs into difficulty. Controlling the whole operation is a blending mechanism which takes the two sets of predictions for letters, blends them together according to the confidence of each of the models, and invokes an arithmetic coder with the resulting probabilities.

The concept of model blending has application even if a parallel text is not available. In the future, for compression methods to improve, techniques from artificial intelligence will need to provide semantic information to be used in conjunction with the present syntactic models to take advantage of the two components of texts. The design decisions made here for parallel compression may well be precursors of those made in the near future for the next generation of text compression techniques.

For the general compression model, a technique called *prediction by partial match* (PPM) was chosen [Cleary and Witten 1984b]. PPM is the best general compression method currently known. In 1988, two variants of this technique won a \$30,000 prize in an Australian compression competition. In our system we use the variant described by Moffat [1988]. PPM bases predictions on a context of the few characters prior to the character being predicted. When a character is to be predicted, PPM looks at the frequencies of characters that have appeared in previous occurrences of the current context. Frequently occurring characters are given high probabilities.

Our model which makes predictions from a parallel text exploits two relationships between parallel text pairs. The first is the case where a word in one text corresponds exactly to the same word in the other text, and the second relationship is where a word corresponds to a synonym of a word in the other text. We ignore more complex relationships, such as a synonym phrase. In the following we refer to the text being compressed as the *target* text, and the source translation available to assist predictions is referred to as the *parallel* text.

Firstly, the parallel model reads a chunk of the parallel text (where a chunk is the smallest portion of the source text for which an exactly equivalent portion can be found in the target text, for example a verse of the Bible or Classical texts) and bases its predictions on this information. Each word in the parallel text is compared to the context of the current position in the target text. If there is a match, the next letter in the word from the parallel text (source word) is noted as a candidate for the next letter in the target text. For example, consider the following target text context for compression. It is a translation from the Roman historian, Tacitus.

Yet the age was not so barren in noble qualities, as not also to exhibit examples of virtue. Mothers accompanied the flight of their sons; wives followed their husbands into exile; there were brave kinsmen and faithful sons in law; there were slaves whose fidelity defied even t

And the parallel text:

However, the period was not so barren of merit that it failed to teach some good lessons as well. Mothers accompanied their children in flight, wives followed their husbands into exile. There were resolute kinsmen, sons-in-law who showed steadfast fidelity, and slaves whose loyalty scorned the rack. Distinguished men driven to suicide... (etc)

To predict the letter following the “t”, the parallel model looks at all the words in the same verse in the parallel text beginning with “t”: “the”, “that”, “teach”, “their”, “their”, “There”, “the”, and “to”. Candidates for the next letter would thus be “h”, “h”, “e”, “h”, “h”, “h”, “h” and “o” respectively. The frequencies of the candidate letters are noted. It might seem a little crude to take words from the whole verse, and of course if sentences were known to correspond (as they do in this case) then the frequencies could be more accurate; in the example, only the words “There” and “the” would be considered.

The same process is applied to the synonyms of every word in the parallel chunk, resulting in another set of frequencies for candidate letters. In the above example, a synonym for “rack” could be “torture”, so the letter “o” would appear in this distribution. (The word is, in fact, “torture”.) The two frequency distributions are added, with less weight being given to the synonym frequencies. The resulting relative frequencies of letters are used as the prediction for the parallel model.

It was found that synonyms had to be used very selectively. Without some discrimination, they were of no help whatsoever, because so many were generated that they swamped any correct word with many other words starting with the same letters. However, once three or more letters of a word in the target text had been seen, and if no words in the parallel text matched the context, it was possible to weed out most of the synonyms and to make useful predictions based on the few that were left.

The parallel model predictions are then combined with the predictions of the standard PPM model. The weighting of the two models was carefully tuned, the main savings being achieved by the selection of a weighting which depended on the number of letters of the current word that have been seen. The usefulness of a parallel text tends to increase as more letters of a word are seen; if a parallel word matches the first four letters of a target word, it is very probable that the parallel word is the correct word, and greater weighting is given to the parallel model. Suitable weights were determined experimentally. Figure 4 shows the weight given to the parallel model depending on how far into a word prediction is. For example, the first letter should be predicted with a 70% weighting in favour of the PPM model, while the fourth letter of a word should be predicted with an 80% weight in favour of the parallel model. These values achieve good compression, and are not very critical—a deviation of 10% or 20% from the values given has very little effect on compression.

Another technique to improve the predictions of the parallel model is to carefully follow the relationship between the word and phrase order in the source and target texts. Our model gives more weight to matching words near what it thinks is the corresponding portion of the source text. The corresponding point in the parallel text is set whenever an exact match for a word is found. After that it moves by one word for every word that is encoded, but the weighting given to such predictions is diminished until another word matches exactly. The weighting given to words in the parallel text is proportional to their distance from what is thought to be the current point. Thus we avoid giving undue weighting to words that are unrelated to the current phrase, and the system is able to follow interchanged phrases.

The resulting compression method was tested on several pairs of texts. Table 2 shows the results of various combinations of target and parallel texts. The texts are the books of Genesis and Esther from the Bible. In all cases the use of a parallel text gave significantly better compression than the PPM coder on its own. The size of the improvement corresponds to the similarity of the versions. For example, Today's English Version and the King James Versions are quite different, and the compression is only 18% better than it would be without the help of a parallel text. An improvement of 40% was obtained in the fourth experiment. The best results achieved by humans are shown in parentheses. The improvement in these cases is better than the computer achieved, which indicates that there is room for improvement in the mechanical predictions. Nevertheless, the compression achieved using a parallel text is considerably better than existing methods.

6 Conclusions

The availability of parallel texts enables an improvement of some 50% on the entropy of the predictions of humans, and up to 40% on automated prediction. This is not as large as might be expected, and so there is apparently a significant amount of information in a text quite apart from the raw semantic information, in the form of style, vocabulary, and subtle changes in connotation.

The 30-40% gain in compression does not seem to warrant the complications of having a thesaurus, machine translator and the parallel compression program. While further saving might be expected, the human experiments indicate that only a little more improvement is likely to be achieved in the future. Nevertheless, this still represents a significant achievement in terms of compression; achieving a small amount of extra compression is harder and harder as the compression gets better. Coding a 28-symbol alphabet in eight bits requires virtually no work. Packing it into five bits per character is somewhat more difficult. With Huffman codes [Huffman 1952], three to four bits per character is achievable. The most sophisticated general compression scheme known manages about two bits per character, and dramatic improvements on this

method are not forthcoming. Yet the relatively sophisticated techniques using a parallel text described here can almost cut this in half again, to 1.2 bits per character, although to go lower will require a much greater amount of work.

Apart from developing and tuning a highly effective compression method for parallel texts, we have made additions and improvements to the gambling method of estimating entropy, and constructed a framework for blending two prediction models, which will be useful in the future for sophisticated compressors that exploit the semantic content of a text.

References

- Bell, T.C. , Witten, I.H. and Cleary, J. G. (1989), “Modelling for text compression”, *ACM Computing Surveys*, 21 (4), 557-591, December
- Bell, T.C. , Cleary, J.G. and Witten, I.H. (1990), *Text compression*, Prentice Hall (Englewood Cliffs)
- Cleary, J.G. and Witten, I.H. (1984a) “A comparison of enumerative and adaptive codes,” *IEEE Trans. Information Theory*, *IT-30* (2), 306-315, March.
- Cleary, J.G. and Witten, I.H. (1984b) “Data compression using adaptive coding and partial string matching,” *IEEE Trans. Communications*, *COM-32* (4), 396-402, April.
- Cover, T.M. and King, R.C. (1978) “A convergent gambling estimate of the entropy of English” *IEEE Trans Information Theory*, *IT-24* (4) 413-421, July.
- Huffman, D.A. (1952) “A method for the construction of minimum-redundancy codes,” *Proc. Institute of Electrical and Radio Engineers*, 40 (9), 1098-1101, September.
- Lelewer, D.A. and Hirschberg D.S. (1987) “Data compression,” *ACM. Computing Surveys* 19(3), 261-296
- Moffat, A. (1988) “A note on the PPM data compression algorithm,” Research Report 88/7, Department of Computer Science, University of Melbourne, Parkville, Victoria, Australia.
- Shannon, C.E. (1948) “A mathematical theory of communication” *Bell System Technical J*, 27, 398-403, July.
- Shannon, C.E. (1951) “Prediction and entropy of printed English” *Bell System Technical J*, 50-64, January.
- Witten, I.H., Neal, R., and Cleary, J.G. (1987) “Arithmetic coding for data compression,” *Communications of the Association for Computing Machinery*, 30 (6), 520-540, June.
- Zunde, P. and Kelman, D. (1985) “An information-theoretical metric for testing program comprehension,” Army Institute for Research in Management Information, Communications, and Computer Sciences, Georgia Tech, Atlanta, GA 30332, August.

Tables and Figures

- Table 1 Texts and results for human prediction experiments. Compression is measured in bits per character.

Table 2	Performance of automated parallel text compression system. Compression is measured in bits per character. Performances of best human subjects are shown in parentheses.
Figure 1	Example of parallel text compression for a different language pair
Figure 2	Layout of screen for gambling experiments
Figure 3	Log file from gambling experiment. The subject has just seen “If it pleases your majesty to grant my h” and is predicting the first few letters of “umble request”
Figure 4	Weighting given to parallel model vs. number of characters into current word

	Source	Predicted text	Parallel text	Entropy (unaided)	Entropy (with parallel text)
1	Bible Esther 7:2-3	Today's English Version	King James Version	0.92, 1.3, 3.4	0.47, 0.71, 0.90
2	Bible Esther 7:6-7	New International Version	Today's English Version	1.5	1.0, 1.1, 1.7
3	Tacitus: The histories 1.9	Hadas Translation	Wellesley Translation	1.5	1.2

Table 1

Text	Target (compressed) version	Source (parallel) version	Compression without aid of parallel text	Compression with aid of parallel text
Genesis	King James	Revised Standard	1.98	1.30
Genesis	Revised Standard	King James	2.05	1.42
Esther	King James	Revised Standard	2.10	1.45
Esther	Revised Standard	King James	2.21	1.32
Esther	New International	Today's English	2.31 (1.5)	2.02 (1.0)
Esther	Today's English	King James	2.32 (0.92)	1.92 (0.47)

Table 2

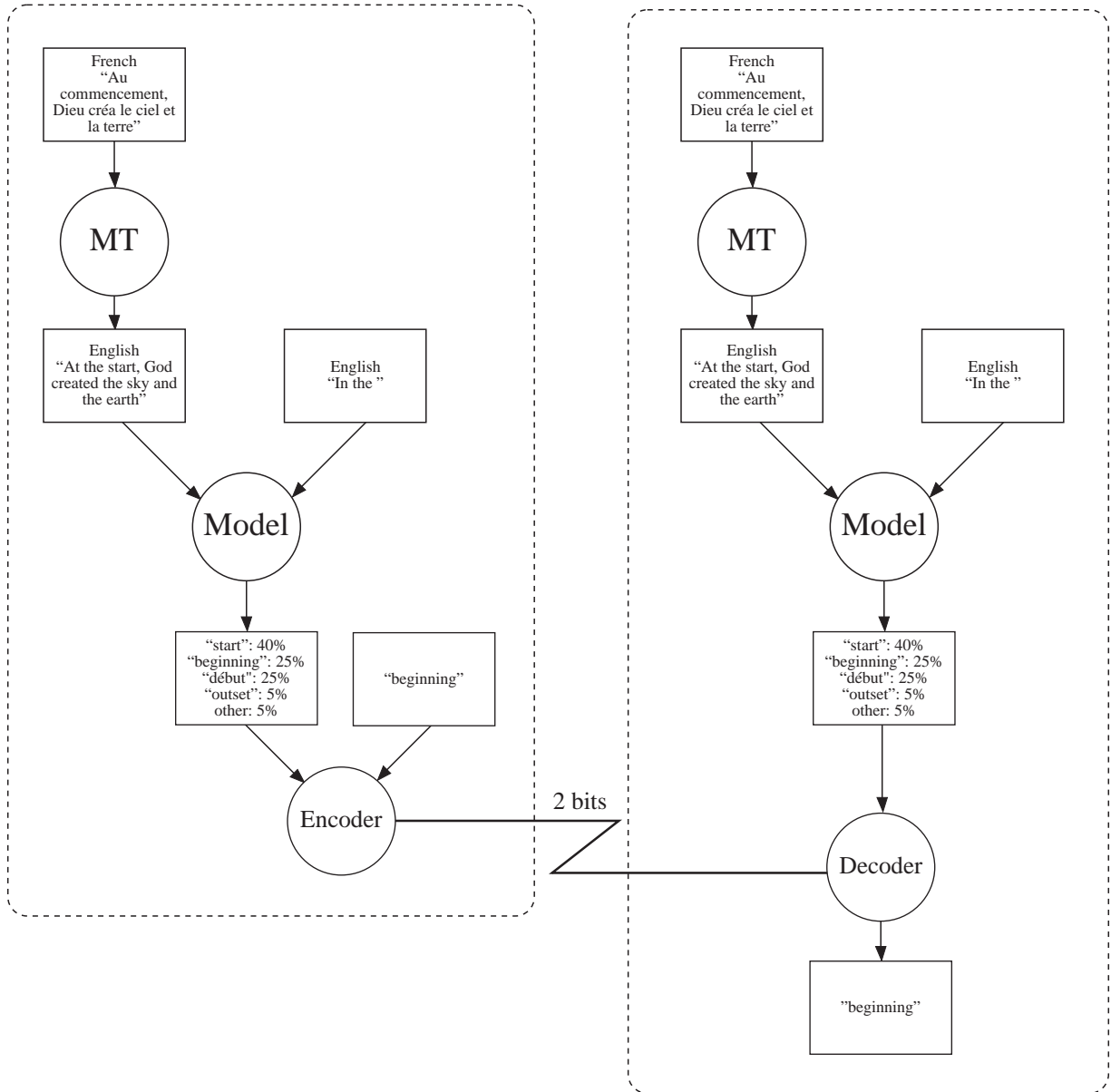


Figure 1

```

At 17:53:09: Char "u"
At 17:55:43: Bet 80% on "aeiou"
At 17:56:55: Bet 25% on "e"
At 17:57:13: Bet 20% on "a"
At 17:57:28: Bet 15% on "i"
At 17:57:45: Bet 13% on "o"
At 17:57:46: Bet 7% on "u"
At 17:57:46: Win
At 17:57:51: Char "m"
At 17:58:55: Bet 70% on "m"
At 17:58:55: Win
At 17:58:59: Char "b"
At 17:59:28: Bet 90% on "b"
At 17:59:28: Win
At 17:59:33: Char "l"
At 17:59:41: Bet 95% on "l"
At 17:59:41: Win
    
```

Figure 2

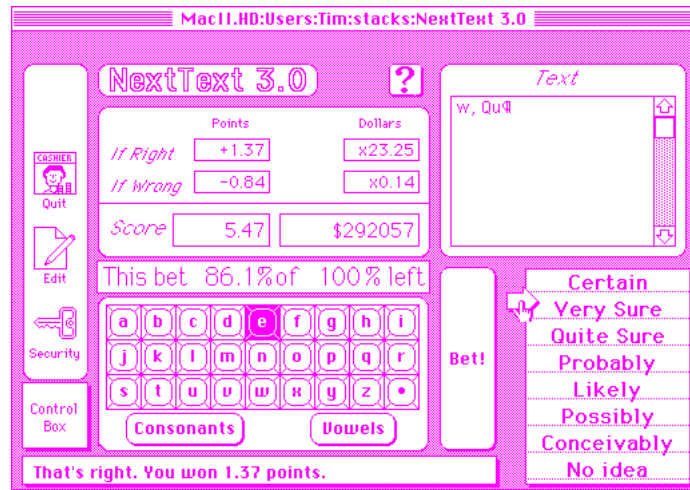


Figure 3

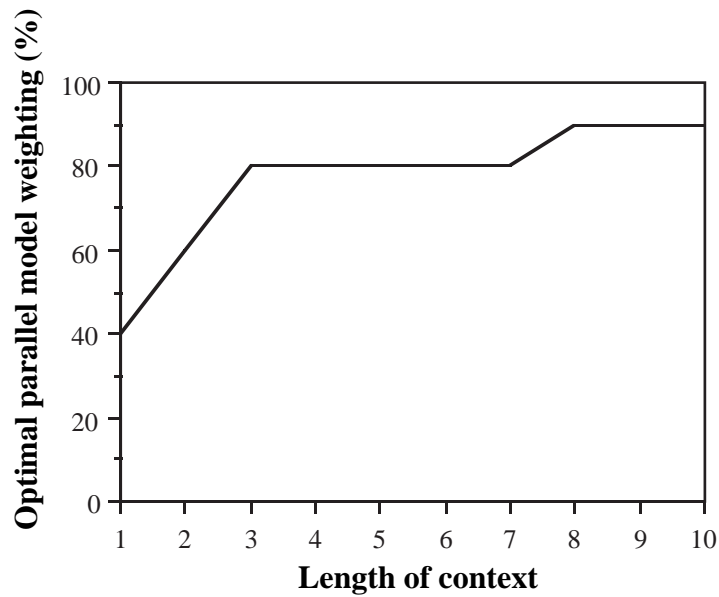


Figure 4