DIVISION OF COMPUTER SCIENCE

Using a neural net to determine the language in which a text is written

Caroline Lyon
Calvin Matthews

Technical Report No.212

January 1995

Using a neural net to determine the language in which a text is written

Caroline Lyon and Calvin Matthews

Abstract

There are statistical patterns of letter sequences in natural language, and different languages have different characteristic patterns. This effect can be used to determine in which language a text is written. The patterns are captured with a single layer, feed forward neural net trained in supervised mode. The sequential dependencies of letters are modelled by taking adjacent letter pairs and letter triples. Training and test data are converted to sets of these tuples, which are the basic elements classified by the network.

This approach is supported by information theoretic results on the entropy of letter sequences for English. The architecture of the network used is shown to be appropriate for data with the characteristics of natural language letter sequences.

For 3 languages over 99% of test strings are correct. For 4 languages, including Dutch and German which are similar, over 92% are correct.

1 Background from information theory

Shannon's well known work on characteristics of the English language examined the entropy of letter sequences [1]. Taking an alphabet of 26 letters he produced a series of approximations H_0, H_1, H_2 ...to the entropy H of written English. These approximations successively take more of the statistics of the language into account, and H_n can be called the n-gram entropy, which measures the amount of entropy with information extending over n adjacent letters of text.

H_0 is taken by definition to be $log_2 26 = 4.7$ bits per letter. This represents the average number of bits required to determine a letter with no statistical information. H_1 is calculated with information on single letter frequencies. If $p(x)$ is the probability of letter x occurring, (x, y) is a tuple and $p(x, y)$ is the probability of the tuple occurring, and the conditional probability of y given x is $p(y \mid x)$ then

$$H_1 = -\sum_{k=1}^{26} p(k) log_2 p(k) = 4.14$$

$$H_2 = -\sum_{j,k=1}^{j,k=26} p(j,k) log_2 p(k \mid j) = 3.56$$

$$H_3 = -\sum_{i,j,k=1,}^{i,j,k=26} p(i,j,k) log_2 p(k \mid (i,j)) = 3.3$$

The derivations of these formulae are in Shannon's original paper and subsequent text books.

The entropy measures can be reduced if an extra character representing a space between words is introduced. Consider the letter sequences in

T W O W O R D S

If the space is included, there are less atypical sequences such as O W O and therefore less uncertainty for H_n where $n > 0$, even though the number of choices has increased. "A word is a cohesive group of letters ...and the n-grams within words are more restricted than those bridging words" [1]. Table 1 gives a comparison of results.

The fact that local dependencies of 2 and 3 letter tuples reduce the uncertainty in letter sequences indicate that bigrams and trigrams hold implicit information on the statistical properties of English. This suggested that different languages could be characterised by sets of bigrams and trigrams.

This hypothesis is also supported by Zipf's earlier work [2] showing that certain morphemes are used frequently, particularly in highly inflected languges like German. Typical letter patterns are likely to occur in a small amount of text. Zipf showed that linguistic elements like morphemes and words have a distinctive distribution. The commonest elements occur very frequently, while many elements occur rarely. It has been shown that the 14 most commonly used words in English account for 30% of all words [3] and yet a modest vocabulary would be 15,000 words. Zipf's empirical law for words in English and other languages gives a relationship between the probability of a word occurring, $p(w)$, and the rank of that word in a frequency table, r.

$$\log p(w) \propto \frac{1}{\log r}$$

Atwell points out that the Zipfian type of distribution may be typical of many linguistic phenomena [4, page 40]. Support for this hypothesis is found in the work described here - see Section 5.

2 The language determination task

If different languages can be detected by the letter patterns in them, a neural network is an appropriate classifying tool to use. One application for such a system could be in support of an automatic tagging system like CLAWS [5]. If a foreign quotation is met in a corpus of English text, the automatic tagger will assume that the words are English and use morphological rules to allocate tags. For a language other than English this is meaningless. If text was run in parallel through a language determiner, this problem could

	H_0	H_1	H_2	H_3
26 letter	4.70	4.14	3.56	3.3
27 letter	4.76	4.03	3.32	3.1

Table 1: Comparison of entropy for different n-grams, with and without representing the space between words

be picked up. Short sequences of foreign text can be automatically detected, as little as 15 characters in the tests described here.

The project described in this paper originally intended to investigate performance levels of different net architectures, single and multilayer feed forward nets operating in supervised mode. However, the results from the single layer net were sufficiently good (see section 5) that the use of multi-layer nets was not pursued. Reasons why single layer nets should be competent for this task have been examined by Lyon [6]. Initially the network distinguished between English and German, but was extended to differentiate between Italian and Dutch too. These languages were chosen because text was easily available on disk.

The texts used are reports written by students in English and German, and Italian text from a computer network information file. Dutch text comes from a story and also from an OS/2 help file.

3 Representing the input

The input is a string of text in English or German. Subsequently this was extended to 4 languages, but the method used remains similar. There is an alphabet of 31 characters, consisting of 26 letters, a space and 4 special characters from German. There is also a start symbol. This alphabet is represented by the integers 0 to 31.

It may be argued that the use of the German special characters makes it possible to perform the classification task without recourse to a neural net. However, their frequency is less than 0.5%. Approximately 1 in 6 of the German strings contain a special symbol. The percentage of strings that were correctly classified was approximately the same for strings with and without the special characters.

Punctuation is not represented, since it usually occurs at word boundaries where a space is equally significant. Apostrophes within a word are ignored.

The string is bound by a minimum length of 15 characters, unless the 15th character is not ar the end of a word; in this case the string is extended to the end of the word.

```
0 16 18 15 7 18 1 13 13 1 20 21 21 18 32
1 language one
0 13 1 14 1 7 5 18 32 3 15 14 20 18 15 12 5 16 1 14 5 5 12 32
1 language one
0 19 5 3 20 9 15 14 32 6 15 21 18 32 1 3 20 9 22 9 20 9 5 19 32
2 language two
0 23 8 9 12 19 20 32 1 20 32 10 1 5 7 5 18 32 20 8 5 19 5 32
2 language two
0 9 14 3 12 21 4 5 4 32 14 21 13 5 18 15 21 19 32
2 language two
0 2 5 23 1 11 5 14 32 5 14 32 1 14 1 12 25 19 5 18 5 14 32
1 language one
0 32 22 9 19 9 20 19 32 20 15 32 15 20 8 5 18 32 19 9 20 5 19 32
2 language two
```

Figure 1: Typical format of a preprocessed file. Strings of text are represented as a sequence of numbers, followed by the desired language classification

The input vectors for the neural network

A string of integers is converted into an input vector in the following way. The elements of the input vector represent all possible pairs and triples, adjacent integers taken together as a unit. Initially, all these tuples are disabled, but when one occurs in a training string it is activated.

Using this representation, a word like QUEEN will be decomposed into tuples QU, UE, EE, EN, QUE, UEE, EEN. A 5-letter word generates 7 tuples, which could appear in other words. Thus, the net is able to generalise from the training data to unseen test data. This example shows intuitvely how the network operates. A sequence like QUE is common in French as well as English, but the sequence EEN is rare in French. The likelihood of the set of all 7 tuples coming from English text will be higher than the likelihood of them coming from French.

These bigrams and trigrams partially capture the sequential nature of the input. They will not be able to represent distant dependencies. However, Shannon's work indicates that pairs and triples capture a significant amount of the implicit information of the letter sequence. Table 1 shows how entropy decreases as longer n-grams are used, but the rate of reduction declines as n increases.

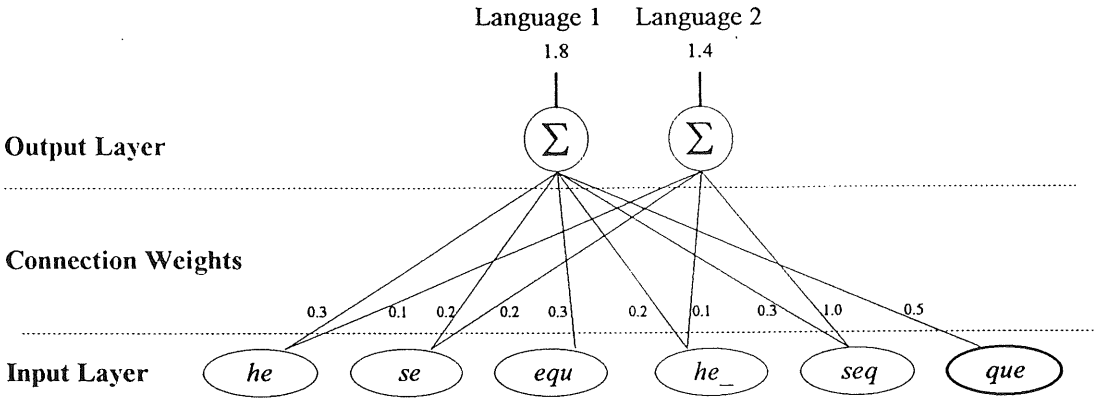

Figure 2: The architecture of Hodyne. The net has a single layer of processing nodes and an input layer. The net is not fully connected: the tuple 'que' is only connected to the output node representing language 1

4 The network architecture

The architecture of the net is simple. It is derived from the original Hodyne net, introduced by Wyard and Nightingale, described in [7], with a learning rule similar to that used in the perceptron.

The input strings are binary vectors, presented to the network one at a time. Each element of the vector represents a letter tuple, and if it is present in the string it will be flagged. These elements are the input nodes. They are linked by weighted connections to the 2 output nodes, which represent "language 1" and "language 2". The number of outputs was later extended to 4.

When a string is presented to the trained net the weighted links carry activation to the 2 output nodes, and these are summed for each of them. The highest scoring node wins.

4.1 The training process

Training strings are presented to the net one at a time. Then connections are made between the input nodes and the desired output node, if they do not already exist. Weights are initially 1.0. Then the weights on the connections to each output node are summed and the one with the greatest activation will fire. If the desired result is obtained the weights are left unaltered. However, if the desired result is not obtained the weights are adjusted. It should be noted that the net is not fully connected. Some letter tuples appear in both languages, others only in one.

When an unacceptable result is obtained the weights on the connections to the wrong answer are decremented by an update factor, while those on the connections to the right answer are incremented. Then the summing process is repeated. The update factor is calculated from a function that has to meet several requirements. It should always be positive, and asymptotic to maximum and minimum bounds. The factor should be greatest in the central region, least as it moves away in either direction. We are currently still using the original Hodyne function because it works well in practice. This is

$$w_{new} = \left[1 + \frac{\delta * w_{old}}{1 + (\delta * w_{old})^4}\right] w_{old}$$

where δ= +1 for strengthening weights and δ = -1 for weakening them. Graphs of the update function are shown in Figure 3, where the top curve is the incrementing function, and the lower the decrementing one.

An incremental method of training has been found most effective. A subset of strings from the training set are taken, and presented to the net one at a time. Each is fed forward and the result for the string is found. The weights are adapted if necessary. If the percentage of strings correct for this subset is below the training criterion specified then the net trains on this set again. When the training criterion is met, then the next group of strings is added to the current training set and the cumulative total is trained. This is continued until the whole net is trained. The training step size has been varied.

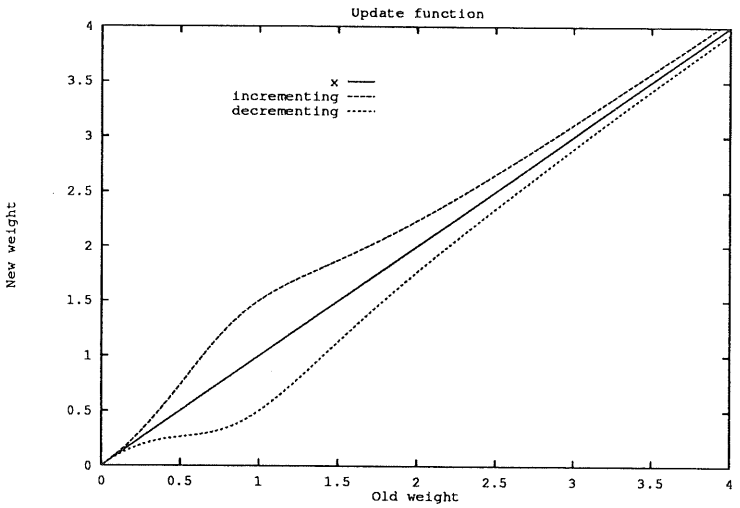

Figure 3: Relationship between old and new weights

4.1.1 Note on implementation

In training the strings from each language are selected at random, with 2 variations. First, the number of consecutive strings from one language cannot exceed 4 while the net is in training mode. Secondly, the minimum string length is limited: very short sentences are discarded in the training phase. While evaluating this prototype similar restrictions are also applied to the test data.

4.2 Testing

Once the net has been trained, weights are fixed and unseen data can be processed. Sentences are decomposed into letter tuples, which are mapped onto the input vector. These inputs are fed forward through the net, and the output node with the highest activation fires. If a letter tuple has not appeared in a training text it will not be connected and will make zero contribution to the result. Occasionally, when a net has been trained on

```
TEST_RESULTS

                WT_Lang 1        WT_Lang 2        Winner
1.              30.764706        19.871134        Language 1
2.              12.764662        23.475702        Language 2
3.  Incorrectly identified string  0 4 9 19 3 21 19 19 5 4 31
    In language 1 actual string - DISCUSSED_
    >>>>>>>>>>  21.000000        23.500000        Language 2
4.              14.500000        31.834433        Language 2
5.  Can not classify the string  0 10 5 4 5 19 31 16 1 1 18 31
    In language 2 actual string - JEDES_PAAR_
    >>>>>>>>>>  22.364161        20.128866        None
6.              21.393551        36.908714        Language 2

Sentences correct: 4 out of 6, 66.666667 percent correct
```

Figure 4: Typical format of output from the net. For correctly classified strings the sum of the connection weights for each node and the classification are displayed (strings 1,2,4,6). Incorrectly classified strings are displayed in both numeric and alphabetic form as well (string 3). Where the net is unable to classify a string it is displayed and counted as incorrect (string 5)

4.3 The learning algorithm

Learning algorithms can be divided into those that update weights whether the desired result is obtained or not, and those that only adjust weights if the desired result is not obtained. Classical Hebbian learning falls into the first category: correct results are reinforced. The perceptron learning algorithm, followed by the Widrow-Hoff method and the back propagation algorithm, only adjust weights when the desired result is not obtained. Hodyne falls into this second category. However, the Hodyne algorithm differs from these others, where an internal error between desired and actual output is calculated. Then the weight adjustment aims to reduce this error to a minimum. Hodyne weight adjustments are based on the existing weight values, not on the internal error measure. This can be compared to the "semantic level error feedback" used by Gorin et al. in "Adaptive Acquisition of Language" [8]. Learning is driven by a penalty function.

5 Results

5.1 Comparison of tuple sizes - Test I

The results described in this section are based on 4 data sets, taken from English and German text. The first 1/3 was taken for training, the remaining 2/3 for testing. Three sets have a minimum string length of 30, and consist of the same data shuffled in different orders, so that the training and test sets vary. The fourth set has a minimum string length of 15, derived from the 2nd set.

These data sets were processed using pairs and triples, pairs only and triples only. As Table 3 shows, performance is slightly better using triples alone. Whereas a number of letter pairs occur commonly in both English and German text, letter triples are more likely to be characteristic of one language alone.

During training the threshold of acceptability was set at 95% and 100%. In both cases the training sets reached 100% correct. In testing the incorrect strings include proper nouns and foreign words - typically English words like "programme" and "manager" in German text.

	Files Used	Min. String Size	Total No. of Strings	No. Training Strings	No. Test Strings
SET 1	Eng1,Ger1	30	1064	354	710
SET 2	Eng2,Ger2	30	1063	353	710
SET 3	Eng2,Ger1	30	1064	354	710
SET 4	Eng2,Ger2	15	1948	649	1299

Table 2: Data sets used for training and testing - Test I

	Pairs and Triples		Pairs Only		Triples Only	
	No. Strings Correct	Percentage	No. Strings Correct	Percentage	No. Strings Correct	Percentage
SET 1	686	96.62	683	96.20	702	98.87
SET 2	697	98.17	684	96.34	700	98.59
SET 3	686	96.62	686	96.62	700	98.59
SET 4	1236	95.15	1198	92.22	1239	95.38

Table 3: Results using pairs and triples, pairs only and triples only

	No. Training Strings	No. Test Strings	No. Strings Correct	Percentage
TEST 5	710	354	352	99.44
TEST 3	354	710	702	98.87
TEST X	177	354	344	97.18
TEST Y	88	354	339	95.76
TEST Z	44	354	340	96.05
TEST A	22	354	335	94.63
TEST B	11	354	329	92.94
TEST C	5	354	264	74.58
TEST D	3	354	240	67.80
TEST E	2	354	206	58.19

Table 4: Results using different size training sets

5.1.1 Zipfian distribution of letter tuples

Inspection of the processing of this data showed that many tuples only occur once or twice, whilst a few appear frequently. When the test sets are presented a significant number of the tuples have not appeared in the training sets. This supports the general hypothesis that Zipfian distribution is typical of many phenomena in natural language.

5.2 Comparison of training set sizes - Test II

These experiments were carried out on the same master data sets, partitioned differently into training and test sets. Only triples were used, as this representation had worked best previously. The results are given in Table 4.

The very slow degradation of performance is interesting. Training on only 11 strings still gives nearly 93% correct. The final 2 training strings were the phrases "involved in a number of activities" for English and "marketingplan der abteilung" for German.

These results indicate the high level of redundancy in the data.

6 Experiments with more languages

The neural net was modified to produce up to 6 outputs so that more languages could be determined simultaneously..

The processor next took in Italian text. The alphabet was extended from 30 to 31 characters, including the apostrophe as a special character. This can occur in all 3 languages, but is commonest in Italian. Preliminary experiments compared the 3 languages taken in pairs. Letter triples only were used, and the threshold of acceptability for training was 95%. Results are given in Table 5.

6.1 Distinguishing 3 languages taken together

A set of 1911 strings of mixed English, Italian and German were prepared. These were first divided with 1/3 for training, 2/3 for testing, and then vice versa. The results are given in Table 6. Once again, the performance is very good.

As before the errors occurred in strings with proper names. "Carlos Simpson" kept incurring an error. Another source of difficulty was again the inclusion of foreign words, such as "input" and "output" in the Italian text.

Languages	No. Training Strings	No. Test String	No. Correct Strings	Percentage
English & German	710	354	352	99.44
English & Italian	756	378	376	99.47
German & Italian	756	378	377	99.74

Table 5: Results on 3 languages taken in pairs

	No. Training Strings	No. Test String	No. Correct Strings	Percentage
SET 7	637	1274	1243	97.57
SET 7	1274	637	632	99.22
SET 8	637	1274	1261	98.98
SET 8	1274	637	632	99.22

Table 6: Results for English, German and Italian together

Languages	No. Training Strings	No. Test String	No. Correct Strings	Percentage
Dutch, English & German	1218	609	594	97.54
Dutch, English & Italian	1499	750	744	99.20
Dutch, German & Italian	1465	733	722	98.50

Table 7: Results for English, German, Italian and Dutch 3 at a time

6.2 Distinguishing 4 languages

This time Dutch was added to the set of languages to be distinguished. Since Dutch is quite closely related to German this tests the processor further. Part of the Dutch text was taken from an OS/2 help file. English text was edited out, but some English words remained. Preliminary results for the languages taken 3 at a time were slightly less good, as shown in Table 7. It was noted that the order in which the languages were presented to the pre-processor affected results. This may be due to a problem with the random number generator, which determines how many adjacent strings of each language are taken into the training set. There is a pattern in the numbers which leads to one language being under represented. However, this should not have a significant effect if redundancy is high.

	Ordering of languages	No. Training Strings	No. Test String	No. Correct Strings	Percentage
SET 13	D,E,G,I	1670	836	780	93.30
SET 14	G,D,I,E	1670	836	770	92.12
SET 15	G,I,E,D	1670	836	798	95.45

Table 8: Results for English, German, Italian and Dutch together

Finally, all 4 languages were taken together. The preprocessor was slightly modified so that a maximum of 2 consecutive strings were taken from any one language. The results are shown in Table 8.

6.3 Processing speeds

The time taken to train and test a data set varies from 8 seconds for 710 training strings in 2 languages, to 40 seconds for 2548 training strings in 3 languages. These times were reduced by an order of magnitude when the net was trained in advance and then saved. The net then only carried out the testing process in real time, taking between 1 and 2 seconds of CPU time.

7 Conclusion

Languages can be characterised by the letter patterns in their texts, and a single layer neural network is an appropriate tool for capturing these patterns.

The network's pattern of connectivity is suitable for linguistic classification tasks as the data seems to have a Zipfian distribution. Since there are a significant number of seldom occurring triples we wish to capture information from some of them. Letter triples that do not occur in the training set make no contribution to the decision on classifying test data. But if a triple occurs once, then its connection weight to the desired output is initialised to

1.0, which can have a significant influence on the outcome. The weights on triples that often occur in more than one language tend to be diminished. Seldom occurring triples can trigger an adjustment to the weights.

Preliminary experiments were made with a 5th language, French, and similar results to those given above were obtained. This indicates that the method of language determination introduced here could possibly be extended.

References

[1] C E Shannon. Prediction and Entropy of Printed English. *Bell System Technical Journal*, 1951.

[2] G K Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology.* Addison Wesley, 1949.

[3] L P Hyvarinen. *Information Theory for Systems Engineers.* Springer Verlag, 1970.

[4] R Pocock and E Atwell. Treebank trained probabilistic parsing of lattices. School of Computer Studies, Leeds University, 1994. In the Speech-Oriented Probabilistic Parser Project: Final Report to MoD.

[5] R Garside. The CLAWS word-tagging system. In R Garside, G Leech, and G Sampson, editors, *The Computational Analysis of English: a corpus based approach.* Longman, 1987.

[6] C Lyon. *The representation of natural language to enable neural networks to detect syntactic structures.* PhD thesis, School of Information Science, University of Hertfordshire, 1994.

[7] P J Wyard and C Nightingale. A single layer higher order neural net and its application to context free grammar recognition. *Connection Science*, 4, 1990.

[8] A L Gorin, S E Levinson, A N Gertner, and E Goldman. Adaptive acquisition of language. *Computer Speech and Language*, 1991.