Tagging Experiments Using Neural Networks

Martin Eineborg and Björn Gambäck[1]
Stockholm

Abstract

The paper outlines a method for automatic part-of-speech tagging using artificial neural networks. Several experiments have been carried out where the performance of different network architectures have been compared to each other on two tasks: classification by overall part-of-speech (noun, adjective or verb) and by a set of 13 possible output categories. The best classification rates were 93.6% for the simple and 96.4% for the complex task. These results are rather promising and the paper compares them to the performance reported by other methods; a comparison that shows the neural network completely compatible with pure statistical approaches.

1. Introduction

Rayner *et al* (1988) experimented with using a formal grammar along with example-sentences to deduce a lexicon. Unfortunately, the exponential explosion that followed from ambiguities in the grammar caused the system to be very slow. The project described in this paper builds on the assumption that one way to attack this problem would be to let a neural network suggest restrictions on the possible word-classes of the unknown word derived from its word-ending and context.

Another motivation for our project is that within the language area connectionist models have so far proved discouragingly unsuccessful compared to other methods. Even though they have been tried out for several applications, such as semantic clustering, preposition choice, etc., the only language area where artificial neural networks have been successfully applied on a larger scale has been speech; however, all currently leading speech recognition systems (the ones in the US DARPA race) have discarded neural nets for Hidden Markov Models, a statistical method. The current state of affairs should however hardly be taken to be the permanent truth. The need for different machine learning methods within the language area should be evident; in this paper we will single out the topic of part-of-speech tagging for special attention, but the last

[1]The work reported here was funded by the Swedish Institute of Computer Science (Asea Brown Boveri, Telefon AB LM Ericsson, Försvarets Materielverk, IBM Svenska AB, NUTEK, and Telia AB). We would like to thank Lars Asker (Stockholm University), Ivan Bretan (IBM), Douglass Cutting (Xerox Parc), Jussi Karlgren (SICS), Pat Langley (Siemens), and Christer Samuelsson (SICS) for helpful discussions and suggestions.

words for other areas such as disambiguation, document matching, information retrieval, grammar and transfer-rule induction, etc., have certainly not been said.

The experiments we have carried out have used different back-propagation network architectures in order to assign part-of-speech tags to unknown words. A brief background to artificial neural networks and the back-propagation algorithm is given in the rest of this section. Section 2 then goes on to describe the different network architectures used in our experiments. The networks were trained on both morphological and (local) context information extracted from a tagged text corpus and then evaluated on previously unseen data from the same corpus. The results of the different experiments are given in Section 3. Section 4 compares these results to other possible methods of solving the problem, i.e., pure statistical and rule-based approaches; finally Section 5 sums up the previous discussions and points to possible future extensions.

Artificial Neural Networks

Several researchers around 1940 suggested that a more brain-like machine should be created. A first step in this direction was taken when McCulloch and Pitts (1943) proposed a model of a neuron, which, just like the biological neuron, takes several inputs and produces one output. The changes in synapses are simulated by weight variables. Modification of the weights is handled by a learning rule. A weight has two features: the sign of the weight determines if the incoming impulse is excitatory or inhibitory and the absolute value of the weight determines to what degree notice should be taken to the incoming impulse. When the incoming values are above a certain level (the threshold) the neuron fires according to a firing rule. In the McCulloch & Pitts model the firing rule can be expressed by the following simple mathematical formula:

the neuron fires iff $\Sigma_k\, x_k\, w_k > \theta$

where x_k is the value received from neuron k
w_k is the weight associated with input from neuron k
θ is the threshold

This model uses only a two-valued output indicating firing, or not. It is still the basis of many neural networks, but has been improved upon several times, in particular when Widrow and Hoff came up with a learning rule called the Widrow-Hoff rule or the delta rule (Widrow 1962).

It can be expressed as:

$$w_k(t+1) = w_k(t) + \alpha\, \delta_k(t) x_k(t)$$

where α is a constant (gain term) typically $0.01 \leq \alpha \leq 10$
$w_k(t)$ is the value of weight k at time t
$\delta_k(t)$ is the error of neuron k at time t
$x_k(t)$ is the incoming value from neuron k at time t.

It was shown by Rosenblatt (1962) that the delta rule causes the weights to converge. He also developed the perceptron, a neuron able to classify binary or continuous valued input into one of two classes; however, a serious blow against neural science came when Minsky and Papert (1969) showed that a perceptron neural network consisting of only one layer is unable to handle nonlinear functions; to do so a hidden layer has to be included in the net. A hidden neuron receives input from other neurons and transmits output to other neurons. A hidden layer consists only of hidden neurons. In 1986 Rumelhart, Hinton, and Williams came up with a network that could handle hidden layers. The method is called backpropagation and will be further described below.

Another model was created by Kohonen (1984/88). It differs from the previous in that it organizes the input data by itself without the correct output pattern being presented, i.e., it uses unsupervised learning. A Kohonen net consists of a number of neurons organized in a two-dimensional plane called a map. The input pattern is given to all neurons at the same time. The neuron for which the Euclidean distance between the input-vector and the weight-vector is a minimum is selected as being the response of the given pattern.

The Backpropagation Algorithm

Backpropagation uses a two-phase learning cycle. During the first phase, the input pattern is propagated through the network. Some sort of distance, usually the Euclidean distance, is calculated between the actual output and the desired output of the net. This distance is the error of the net. The second phase starts with the error being propagated backwards through the net, adjusting the weights along its way. Then the next pattern can be processed. This cycle, called an epoch, continues until the net satisfactory has learnt all patterns, the weights are then frozen and need not be altered. The neurons used differ from those of McCulloch and Pitts in that real values are used as weights, thresholds, and outputs. The output of the neuron is given by:

$o_m = 1 / (1 + \exp\{a_i\})$

where $a_i = \Sigma_j (w_{ij} * x_{ij}) + \theta_i$ is the activation of the i:th neuron.
and w_{ij} is the j:th weight of neuron i
x_{ij} is the j:th input to neuron i
θ_i is the threshold of neuron i.

There are two weight adjustment rules:

for output neurons the error: $\delta_{pj} = (\theta_{pj} - o_{pj})\, o_{pj}\, (1 - o_{pj})$
for hidden neurons the error: $\delta_{pj} = (\Sigma_k\, \delta_{pj} w_{kj})\, o_{pj}\, (1 - o_{pj})$

2. Test Set-ups

A large number of backpropagation network architectures were tested. This section will describe how the net-input was encoded and the actual architectures of the different networks used in the experiments.

Encoding of Network Input

In the text below we will need to use several character sets, e.g., Alphabet_1 and Alphabet_2 respectively defining the Swedish and ASCII alphabets, sets for Swedish vowels and consonants, and some morphologically and phonologically motivated subsets of these. When defining the mappings of the network inputs, we will also need to discuss a particular type of vectors, namely binary vectors of different length with only one 1. These will be referred to as Bin_n where n is the number of digits in the vector. Strings of characters, lexemes, will be subindexed according to what alphabet the included characters belong to.

To represent the encoding of letters, we will introduce five functions which informally can be said to map the character sets above onto the binary vectors Bin_n and perform the following tasks: f_1 simply divides Alphabet_1 into vowels and consonants; f_2 further subdivides the consonants by phonetic category, that is into plosives, fricatives, laterals, trills, and nasals; f_3 is like f_2, but the vowels A and E are singled out from the others, since they behave rather in a special way when inflection is performed; while f_4 and f_5 encode the entire Swedish and ASCII alphabets, respectively.

For the encoding of grammatical categories we will introduce five other functions mapping from the lexemes to the binary vectors, thus: h_1 splits Lexeme_1 into nine categories: nouns, adjectives, verbs, pronouns, determiners, adverbs, prepositions, conjunctions, and infinitival markers; h_2 adds two more categories, one for auxiliaries and one for sentence delimiters; h_3 is like h_1, but with special categories for auxiliaries, idiomatic expressions, and present and past participles. It also splits the conjunctions into subordinating and coordinating ones; h_4 further subdivides the adjectives by comparative form (i.e., positive, comparative, and superlative) and the adverbs by type (normal, comparative, superlative, and comparison); finally, h_5 does for Lexeme_2 what h_1 does for Lexeme_1, but with extra categories for names, numbers, characters, and sentence delimiters.

Network Architectures

All backpropagation networks were three layer architectures consisting of an input layer, a hidden layer, and an output layer. Information was given in localized form. In order to examine the feasibility of the approach, the sizes of the networks were initially kept at moderate levels to increase only gradually. Two information sources were used: the internal structure of the lexeme and N-grams. An N-gram refers to the grammatical categories of N-1 neighbouring words, so we will use 1-gram to refer to the word itself, a 2-gram (here) denotes the word itself and the word to the left, a 3-gram denotes a 2-gram and the word to the right, and so on. When combining the two information sources the vectors were simply appended. The resulting vector was then fed to the network.

All networks in this paper were trained and tested using the Teleman corpus (Teleman 1974). This text consists of almost 80000 tagged Swedish words gathered from a wide range of different genres. The training could be very time consuming, but fortunately for the most part the networks converged rapidly. Typically, only a few epochs were needed until a satisfactory performance was reached. The small number of epochs needed is very likely a result of the text used for training. Since it contains many duplicates, most input patterns were seen and trained several times during one epoch. The training continued as long as seemed reasonable or as long as the performance did not decrease when evaluated on previously unseen material.

TABLE 1: Summary of the network setups for the experiments

Net	Gram (N)	Category-function	Letter-functions 6	5	4	3	2	1	Training epochs	examples
<18,5,3>	3	h_1	-	-	-	-	-	-	2000	5000
<26,5,3>	3	h_1	-	-	f_1	f_1	f_1	f_1	2000	5000
<42,20,3>	3	h_1	-	-	f_2	f_2	f_2	f_2	2000	5000
<44,20,3>	3	h_1	-	f_1	f_2	f_2	f_2	f_2	2000	5000
<52,20,3>	3	h_1	-	f_1	f_3	f_3	f_3	f_3	2000	5000
<73,20,3>	3	h_1	-	f_1	f_3	f_3	f_3	f_4	2000	5000
<136,20,3>	3	h_1	-	f_1	f_4	f_4	f_4	f_4	2000	5000
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	2000	5000
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	2000	7500
<165,20,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	1000	10000
<165,40,3>	3	h_1	f_1	f_4	f_4	f_4	f_4	f_4	100	7500
<169,20,3>	3	h_2	f_1	f_4	f_4	f_4	f_4	f_4	100	10000
<204,40,3>	3	h_3	f_4	f_4	f_4	f_4	f_4	f_4	50	20000
<212,40,3>	3	h_4	f_4	f_4	f_4	f_4	f_4	f_4	100	20000
<282,80,13>	3	h_5	-	-	f_5	f_5	f_5	f_5	50	30000
<295,80,13>	4	h_5	-	-	f_5	f_5	f_5	f_5	50	30000
<423,80,13>	4	h_5	f_5	f_5	f_5	f_5	f_5	f_5	150	30000

Table 1 describes each network in some detail. The number of neurons of a specific network is indicated by a triple <I,H,O> where I is the number of neurons in the input layer, H the same for the hidden layer, and O for the output layer. The other columns of the table define mapping functions, indicate the number of training epochs, etc. Thus the first net, for example, is called <18,5,3>, since it had 26 neurons in total. It used 3-grams only, so its single source of information was that of the context. The grammatical category mapping used, h_1, was very simple distinguishing only between nouns, adjectives, verbs, pronouns, determiners, adverbs, prepositions, conjunctions, and the infinitival marker. Note that no information at all was extracted from the unknown word. As shown in the table, it was trained for 2000 epochs on a text consisting of 5000 examples.

The other nets combined the two information sources available by also inspecting the letters of the unknown word. In order not to make the networks unnecessarily large the mapping between the actual letter and its representation was kept as simple as possible. At first letters mapped onto one of only three classes: vowels, consonants, or $\emptyset$, the latter indicating the lack of any input character in a specific position. This letter-classification was refined first by subdividing the consonants (plosives, fricatives, laterals, trills, and nasals) and later on by separating the letters A and E from the other vowels. Some nets (like <165,40,3>) were included in order to examine if the result would improve with a larger hidden layer, while other nets (as <169,20,3>) mapped the 3-grams differently, for example with the h_2 function which separates the auxiliary verbs from the domain ones and also recognizes sentence delimiters, enabling the tagger to categorize the first and last words of a sentence.

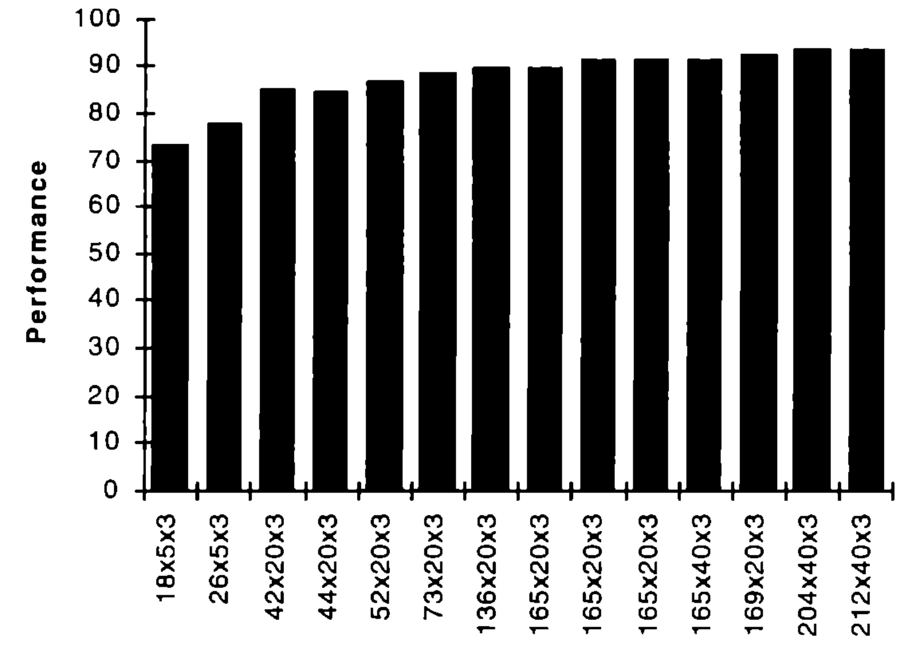

FIGURE 1: Peak performance of the nets on the simple task

3. Results

The networks were tested using an unseen part of the Teleman corpus. The corpus consists of several different types of text. Thus the results should be as general as possible. Figure 1 shows the performance of the nets on the first classification task, part-of-speech categorization. The network with the worst result was not surprisingly the <18,5,3> one, which only used 3-grams. It reached a classification rate of about 73% which is not so bad considering that it extracts no information at all from the word that is to be categorized. When information was added about the internal structure of the unknown word the networks performed better. The more detailed this information was the better did the network perform. The amount of examples used for training was also a parameter that varied. Generally, the more examples that were available to the network the better it performed. The networks with the best results were nets <204,40,3> and <212,40,3>. They both reached a classification rate of 93.6%.

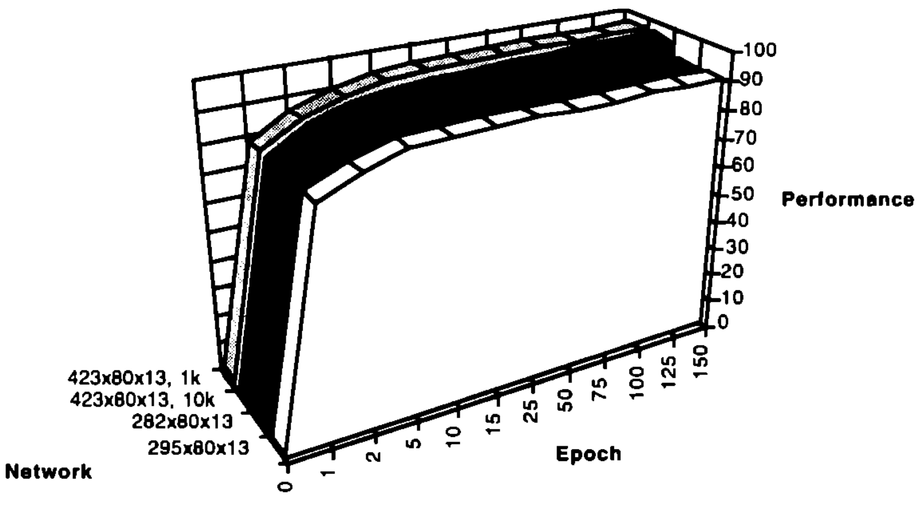

FIGURE 2: Performance of the nets on the complex task

The best result for a network which could classify more than nouns, adjectives, and verbs was 96.4% as shown in Figure 2. This was achieved by the <423,80,13> network, when trained using 30000 examples and tested (like all the other nets) on 1000 unseen examples. To evaluate the consistency of these figures, this net was also tested on an uncommonly large set of 10000 unseen examples. As could be expected when comparing the sizes of the training versus the test sets, this gave a slight decrease in performance, with a top result of 95.80%, as shown by the graph called "<423,80,13>, 10k".

Table 2 shows an example of network outputs. The clause *"(.) i södra Asien (har)"* ["(.) in Southern Asia (have)"] was fed to the <295,80,13> net together with the tags (following the ">" sign). As can be seen from the name *"Asien"*, it had a difficult time separating names from ordinary nouns.

TABLE 2: Example of network output

Categories:	noun	adjective	verb	preposition	adverb	determiner
pronoun	character	conjunction	number	name	sent. del.	inf. mark
Output (right):	0.000000	0.000000	0.000000	0.999984	0.000000	0.000001
0.000014	0.001059	0.000121	0.000000	0.000009	0.000042	0.000000
Right answer:	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pattern:	.>IP I>PR SÖDRA>POSU					
Output (right):	0.000000	0.939394	0.000104	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000138	0.000002	0.000000
Right answer:	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Pattern:	I>PR SÖDRA>POSU ASIEN>PN					
Output (wrong):	0.999468	0.000000	0.000000	0.000000	0.000000	0.325023
0.000000	0.000031	0.000001	0.000000	0.000000	0.000000	0.000000
Right answer:	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
Pattern:	SÖDRA>POSU ASIEN>PN HAR>HVPS					

4. Discussion

In this section we will try to compare the results of the previous section with those that have been obtained using statistical and rule-based methods. First, however, we note that Veronis & Ide (1990) used an approach akin to a neural network in extracting lexical information from a machine readable dictionary. Their results were rather discouraging, in that they managed to identify the correct sense of a word (that already occurred in the dictionary) in only 71.74% of the cases. Nakamura *et al* (1990) investigated word category prediction using a neural network architecture called NETgram, a four layer architecture based on backpropagation. The grammatical categories of the preceding words were used to predict the category of the next word. They reported a word recognition rate of about 68%.

Recently a rule-based approach has achieved some extraordinary results (Voutilainen *et al* 1992). They report a classification rate of 99.7%. The downfalls of their method (and all rule-based ones) are that it is very time consuming to develop the rules and the system produced is highly language dependent. The main objection to their method is however that it also demands a very large lexicon (again making the approach highly language specific). The lexicon they used covered about 95% of all lexemes appearing in the texts, making the comparison of performance figures somewhat unfair.

Samuelsson (1994) suggests a method based purely on statistical evidence. With a success rate of 95.38%, it does not do as well as the method Voutilainen *et al* use, but on the other hand no external lexicon is needed and no language specifics are assumed. The best result was reported using a 4-gram, inspection of 6 letters, and syllable information. The test setting closely resembles that of the <423,80,13> net above, which reached a classification rate of 96.4%. For the same task the Xerox Parc system "Tagger" (Cutting *et al* 1992) based on a Hidden Markov Model (HMM) method also was able to classify 95% of the words correctly (Cutting 1994). Even though this comparison thus shows the neural net approach ahead by a margin, it indicates that the methods are virtually equivalent for the task at hand.

5. Conclusions and Future Work

We have described a series of experiments where different three-layered back-propagation network architectures were used for the task of recognizing unknown words for a natural language system. Two main tasks were performed: in the first the nets were to classify words by overall part-of-speech (noun, adjective or verb) only, while the second task involved a larger set of 13 possible output categories. The best results for the simple task were obtained by networks consisting of 204-212 input neurons and 40 hidden-layer neurons, reaching a classification rate of 93.6%. The best result for the more complex task was 96.4%, which was achieved by a net with 423 input neurons and 80 hidden-layer neurons. The results are overall rather promising and they are completely compatible with those achieved by purely statistical methods; however, they are still inferior to those reported by a rule-based approach, albeit on a somewhat different task.

A possible way to improve on the results could be to combine several networks, for example have we done some initial experiments using a self-organizing map of the Kohonen type. The idea was to use this map to transform the letters of the unknown word to the two dimensional map and then feed the coordinates of this map to a backpropagation network together with the grammatical categories of the surrounding words; however, this approach has not been very successful - yet. Early results indicate that this combination does not perform better than the backpropagation network which only used 3-gram. The map failed to capture the structure of the words. This approach is still being investigated though.

References

Cutting, D. 1994. *Porting a Stochastic Part-of-Speech Tagger to Swedish.* In Eklund (ed), *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Cutting, D., J. Kupiec, J. Pedersen and P. Shibun. 1992. *A Practical Part-of-Speech Tagger* pp 133-140, *Proceedings of the 3rd Conference on Applied Natural Language Processing*, Trento, Italy.

Eklund, Robert. 1994. (ed) *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Ide, N.M. and J. Veronis. 1990. *Very Large Neural Networks Word Sense Disambiguation.* pp 366-368, *Proceedings of the 9th European Conference on Artificial Intelligence*, Stockholm, Sweden (also as pp 389-394, *Proc. 13th International Conference on Computational Linguistics*, Helsinki, Finland, Vol. 2).

Kohonen, T. 1984/1988. *Self-Organization and Associative Memory.* Springer-Verlag, Heidelberg, Germany.

McCulloch, W. S. and W. H. Pitts. 1943. *A Logical Calculus of the Ideas Imminent in Nervous Activity.* pp 115–133, BULLETIN OF MATHEMATICAL BIOPHYSICS, Vol. 5.

Minsky, M. and S. Papert. 1969. *Perceptrons: an Introduction to Computational Geometry.* MIT Press, Massachusetts.

Nakamura, M., K. Maruyama, T. Kawabata and K. Shikano. 1990. *Neural Network Approach to Word Category Prediction for English Texts.* pp 213-218, *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland, Volume 3.

Rayner, M., Å. Hugosson and G. Hagert. 1988. *Using a Logic Grammar to Learn a Lexicon.* pp 524-529, *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary. Also available as *SICS Research Report - R88001*, Stockholm, Sweden.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanism.* Spartan Books, New York.

Rumelhart, D.E., G.E. Hinton and R.J. Williams. 1986. *Learning internal representations by error propagation.* PARALLEL DISTRIBUTED PROCESSING, Vols. 1 and 2, The MIT Press, Cambridge, Massachusetts.

Samuelsson, C. 1994. *Morphological Tagging Based Entirely on Bayesian Inference.* In Eklund (ed): *Nodalida'93 – Proceedings of '9:e Nordiska Datalingvistikdagarna', Stockholm 3-5 June 1993.* Stockholm.

Teleman, U. 1974. *Manual för grammatisk beskrivning av talad och skriven svenska* (in Swedish). Studentlitteratur, Lund, Sweden.

Widrow, B. 1962. *Generalization and information storage in networks of ADALINE neurons.* SELF-ORGANIZING SYSTEMS, Spartan Books, New York.

Voutilainen, A., J. Heikkila and A. Anttila. 1992. *Constraint Grammar of English.* Publication #21, Department of General Linguistics, University of Helsinki, Helsinki, Finland.