Sentence generation and neural networks

Kathrine Hammervold
University of Bergen
Sydnesplass 7
N-5007 Bergen, Norway
kathrine.hammervold@lhs.be

Abstract

In this paper we describe a neural networks approach to generation. The task is to generate sentences with hotel-information from a structured database. The system is inspired by Karen Kukich's ANA, but expands on it by adding generality in the form of language independence in representations and lexical look-up.

Introduction

In the growing field of intelligent communication (web-browsers, dialogue systems, etc.) the need for a flexible generator has become more important (e.g. Hovy & Lin, 1999). NLG is usually seen as a two-stage process where the planning component takes care of the inter-sentential content planning, while the surface realisation component transforms the content representation into a string of words. Interactions between the two components have called for the micro-planning stage to be postulated in the middle, but still the rule-based pipeline architecture has problems with sequential rules and their two-way relations. Statistical approaches have been developed, and seem to provide flexibility to generation tasks.

The approach taken in this thesis, however, explores generation as a classification task whereby the representation that describes the intended meaning of the utterance is ultimately to be classified into an appropriate surface form. Although the task as such is a complex one, the approach allows its decomposition into a series of smaller classification tasks formulated as input-output mappings rather than step-wise rules. One of the goals of the thesis is to study the ways generation could be broken down into suitable sub-classification tasks so as to enhance flexibility in the generation process in general.

Artificial neural networks are a classification technique that is robust and resistant to noisy input, and learns to classify inputs on the basis of training examples, without specific rules that describe how the classification is to be done. There is not much research into using ANN's for generation, the main reason being long training times. Two notable exceptions are Kukich (1987) and Ward (1997), both argue in favour of NN's robustness, but at the same time point out problems with scalability. We believe that with improved computer facilities that shorten the training time, this new way of looking at generation as a classification task constitutes an interesting approach to generation. We have chosen Kukich's approach, as our application domain is to generate utterances from structured databases.

This paper is structured as follows; we first discuss the general model. The second part briefly describes neural networks. We continue with describing a possible implementation of the model, and finally we draw some conclusions and point to future challenges.

1 The model

The task chosen for the system is to generate sentences with information about hotels. The information is presented in a structured way in a database. It is assumed that certain features and values are given as input to the system.. The system's task is then to generate a syntactically (and semantically) well-formed sentence as a response to some user request, based on the

information it gets from the database. These are some example sentences the model will be able to generate:

The hotel Regina has twenty rooms.
The hotel Regina is a small hotel.
The hotel Regina has thirty single rooms
The hotel Regina is an expensive hotel.
The hotel Regina is an expensive hotel in the city center and the single room price is 4000 BEF.
A single room costs 2000 BEF.

In Karen Kukich's stock-reporter system ANA the task is divided into two parts, represented by two neural networks, one sememe-to-morpheme network and one morpheme-to-phrase network. Facts about the status of the stock market were given to the network as semantic attributes. The network was trainde to map eight possible semantic attributes onto morphemes marked for semantic attributes. These morphemes were then linearized in the second network.
The output of Kukich's first net is a set of English morphemes. The actual morphemes are present on the output nodes of the net (72 possible morphemes). This makes the whole system both language dependent and difficult to modify. In order to add a morpheme to the list of possible morphemes the whole network must be modified.

In order to introduce more flexibility, the task could be broken into a language independent and a language dependent task. The database facts are not dependent on which language they are represented in. Instead of letting the output nodes stand for actual morphemes they could represent the different roles the elements can play in a sentence. Part of the output will stand for the subject or theme of the final sentece. The mental space theory of Fauconnier (1985) provides an attractive way of presenting the relations between the facts without commiting oneself to a particular language. The output of Neural Network I (NN I) will therefore be interpreted as an event space, or template, that can be used for language dependent generation. The actual values on the output nodes refer to a concept stored in a lexicon.

There is also a discourse model in the system, which receives information about which features in the database are relevant for the next sentence. It also determines the level of generalisation required for the output. Consider the two following sentences:

A double room costs 3000 BEF and a single room costs 5000 BEF in hotel Regina.

Hotel Regina is expensive.

Saying exactly how much something costs and saying how expensive or cheap it is, is just two ways of communicating the same thing. Both sentences are based on the same facts. Which sentence is chosen may depend on the level of specificity required. This problem of synonymy is present in various ways in all language generation systems. Kukich has an example of two phrases in her system corresponding to the exact same semantic values (or "sememes"). To get around the problem she added two extra sememes to the input and assigned them random values. In this model we have also introduced an extra input category, but it serves as a feature telling the network whether to output the general or the specific sentence.

In Kukich's network the task of the second network is the ordering of the morphemes into a syntactic string. The second network in this model will also have to order the concepts and map them onto forms that can represent real words in a language. We now also have the advantage of having the event space to help the network generate the real sentence.

In English sentences there are phenomena such as agreement that may span over several words in a sentence. The target sentences above show agreement between subject and verb, in other languages e.g. Norwegian there is agreement between adjectives and the nouns they modify. There have been experiments using neural nets to recognise relationships based on constituency and syntactic structure in sentences. Elman (1989 and 1990) has shown that neural nets can learn to represent the concept of word,

constituency and structural relations by introducing the concept of time into the processing of the representations. Elman takes as a starting point a neural networks architechture first described by Jordan (1986). The novel approach is to represent time by the effect it has on processing and not as an additional dimension on the input. The solution consists in giving the system memory. This is done by introducing a fourth type of units, called context units, *into the network apart from the typical input, hidden and output units. At each cycle the hidden unit activations are copied onto the context units. In the following cycle the context nodes combine with the new input to activate the hidden units.*

Elman poses the problem of whether a network can learn underlying aspects of sentence structure from word order. For the simulation he used 15 different sentence templates capable of being filled with nouns and verbs. He used six different classes of nouns (human, animate, etc.) and six different classes of verbs (transitive, intransitive etc.). The network was trained on random two or three word sentences. The network was supposed to learn to predict the order of successive words. For any given sequence of words there are a limited number of possible successors. It is impossible to know with absolute certainty which word follows the previous, but generalisations can be made based on type of verb (e.g. a noun should not be expected to follow an intransitive verb). The performance of the network was therefore measured according to the expected frequencies of occurrence of possible successors, not which word in reality occurred.

On the basis of the training the network developed internal representations which reflected the facts about the possible sequential ordering of the inputs.

> *"The network is not able to predict the precise order of words, but recognizes that (in this corpus) there is a class of inputs (namely, verbs) that typically follow other inputs (namely, nouns). This knowledge of class behavior is quite detailed, form the fact that there is a class of items which always precedes "chase", "bread", "smash", it infers that the large animals form a class."*
> (Elman 1990, p. 199)

He also succeeds in representing agreement between subject and verb, even in sentences like:

Dog [who chases cat] sees girl

This method of teaching a network the relation between different words in a sentence could also be exploited for language generation. The network can be trained on possible sentence structures and agreement between the elements in the sentence. As a starting point the sentence types of the example sentences above could be used. In a symbolic system they could be represented by the following phrase structure rules, depending on the language in question:

$S \rightarrow NP\ VP$

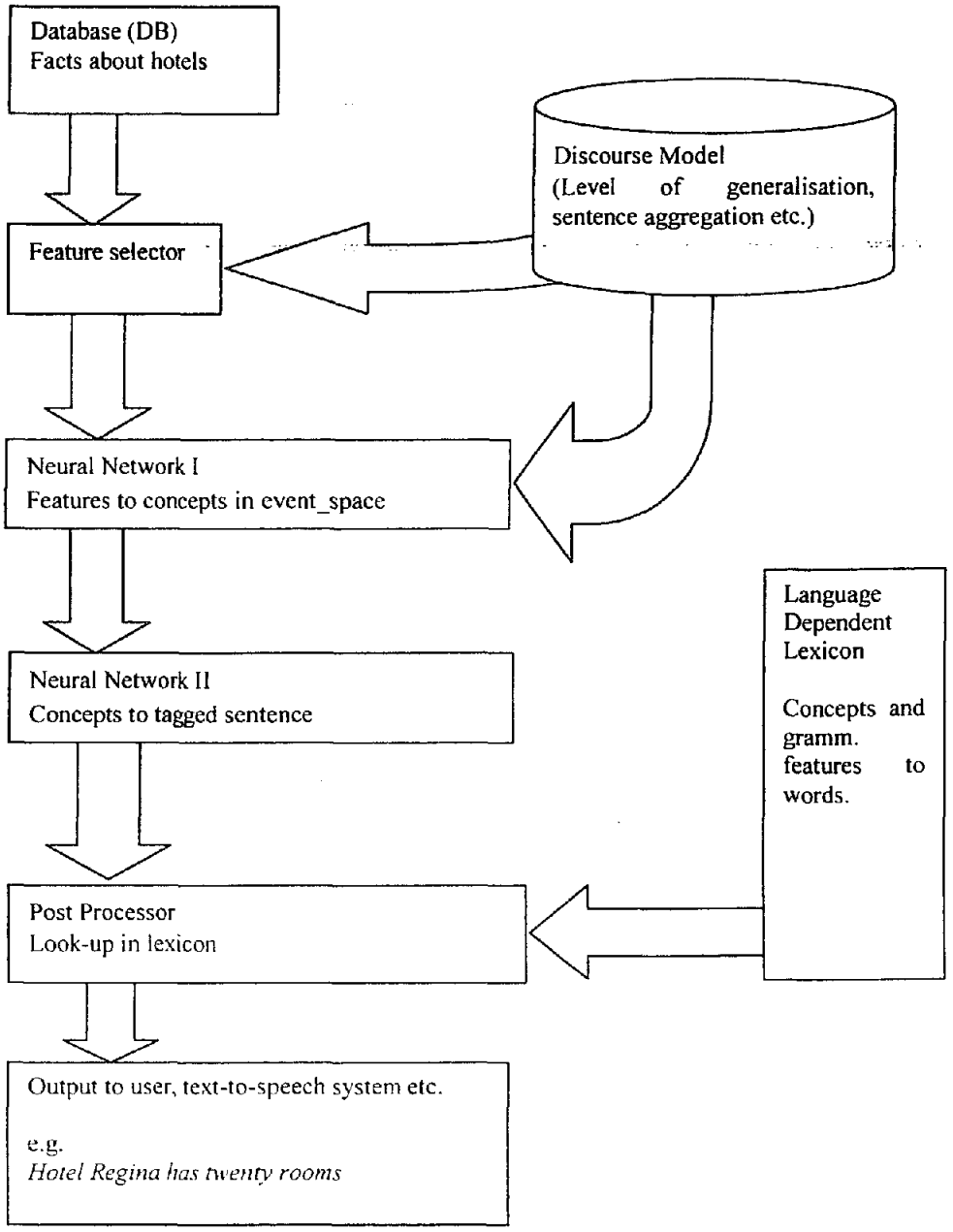
$NP \rightarrow DET\ (MOD)\ N$

$VP \rightarrow V\ NP$

$PP \rightarrow P\ NP$

Each of the categories (N, P etc.) will be output nodes of NN II according to the linear order they may occur in the language in question, and in addition there will be placeholders for number on nouns, verbs and modifiers. The output of NN II is now a linear structure where we know the phrase types. This information could e.g. be used by a text-to-speech system to assign stress to certain word etc.

Our model can now be represented like this:



2 Brief introduction to neural networks

An artificial neural network is a metaphor of the way human and other mammalian brains work. The biological brain consists of a great number of interacting elements, *neuron*. A neuron collects electrical impulses inside the cell membrane. If the level of impulses reaches a certain threshold, the neuron will generate an action potential, a pulse that travels along a thin fibre to other neurons, causing them to store the electrical impulse. Each neuron may have synaptic connections to thousands of other neurons.

An artificial neural network consists of *nodes* or *units*, modelled on neurons. These nodes can receive and transfer *activation*. Each node is connected to many other nodes, and the strength or efficiency of each connection is determined by a *weight*. Together with the nodes the connections are the most salient features of a neural network. The weights on the connections can be modified by *learning rules*, enabling the network to learn new tasks. There are several types of learning algorithms that may be used to train neural networks, but back propagation is probably the most common one. During training, an input/output pair is presented to the network. After a whole set of input/output pairs have been run through the network the back propagation algorithm calculates how far the actual output of the net is from the desired output, and the weights on the connections adjusted in the right direction. If there is any overall pattern to the data, or some consistent relationship between the inputs and results of each record, the network should be able to eventually create an internal mapping of weights that can accurately reproduce the expected output. Since the knowledge the network acquires is a result of the mappings, how the input and output is represented is of great importance.

3 Implementation

The following features are used to describe the information in the database:

Feature	Possible values	Value type	# units in input vector
Service_domain[1]	Hotel	Binary	2
Name	Ariadne, Rabbit, Regina	Binary	2
#_single_rooms	5-50	Numerical value	1
#_double_rooms	5-50	Numerical value	1
Single_room_price	1000-4000	Numerical value	1
Double_room_price	2000-6000	Numerical value	1
Location	City center / Business Park	Binary	2

The feature selector fetches the necessary values (determined by the discourse model) and inputs them to NN I. The input vector is eleven units long. Ten units are the local representations of the features in the database and the last unit represents the generalizer feature from the discourse model. *The Stuttgart Neural Networks Simulator* (SNNS[2]) which will be used for the implementation only allows values between −1 and 1, so the numerical values will be normalized to fit into the vector. This is also necessary so the relative importance of the different features are not out of proportion.

The event space in the output will consist of the following elements:

(*see table 1 at the end*)

The vocabulary needed for the generation task is represented by binary codes, e.g. based on the alphabetical order of the forms. If we let the subject/theme part of the vector be 7 units long

[1] At the moment we deal only with hotel info.

[2] http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html

we can represent 2^7 (128) different word with numerical values. 0000001 is concept number 1, 0000010 is concept number 2 and so on.

Binary code	List of concepts
0000001	ADRIANE
0000010	BEF
0000011	BIG
0000100	CHEAP
0000101	CITY CENTER
0000110	COST
0000111	DOUBLE_ROOM
0001000	EXPENSIVE
0001001	BUSINESS PARK
0001010	HAVE
0001011	HOTEL
0001100	PRICE
0001101	RABBIT
0001110	REASONABLE
0001111	REGINA
0010000	ROOM
0010001	SINGLE_ROOM
0010010	SMALL

In table 2 are some example inputs and outputs, a 1 represents activation on an input or output node.

Now that we have a language independent representation of the sentence we would like to generate, it needs to be cast into a sentence in a natural language. The languages of this system will be English and Norwegian, but the intention is that other languages may also be represented. These input-output combinations shown above should ultimately correspond to the following target sentences (after NN II and post processing):

1) The hotel Regina has twenty single rooms.
Hotell Regina har tjue enkeltrom.

2) The hotel Regina is a small hotel.
Hotell Regina er et lite hotell.

3) A single room costs four thousand Belgian francs.
Et enkeltrom koster fire tusen belgiske franc.

4) A double room costs seven thousand Belgian francs and a single room costs four thousand Belgian francs.
Et dobbeltrom koster syv tusen belgiske franc og et enkeltrom koster fire tusen belgiske franc.

5) The hotel Regina is expensive.
Hotell Regina er dyrt.

6) The hotel Ariadne is a cheap hotel in the city centre.
Hotell Ariadne er et billig hotell i sentrum.

The whole output vector is shown in table 3.

NN II must be trained on agreement. This is done by teaching it to discover relationships, such as the fact that the feature SINGULAR on the subject noun, is associated with the feature SINGULAR on the main verb. The input nodes on Network II will be similar to the output of the first net, but the input will be fed sequentially to the network (theme, number, main_event, complement etc.

If we assume that the output of NN I now serves as the input for NN II, this will be our desired output (only the activated nodes are shown here):

I1: REGINA^SG^HAVE^20^SINGLE_ROOM

O1:
REGINA^DEF^SG^HAVE^SG^20^PLUR^SINGLE_ROOM^INDEF^PLUR.

After post-processing: *The hotel Regina has twenty single rooms*

I2: REGINA^SING^SMALL^HOTEL^SING

O1:
REGINA^DEF^SING^BE^SING^SMALL^SING^INDEF^HOTEL^INDEF^SING

After post-processing: *The Hotel Regina is a small hotel*

and so on...

After a look-up in an English dictionary we find that the singular form of BE is *is*, and the plural form of SINGLE_ROOM is *single rooms*. The reason we do not output this directly is that we would then require different output nodes for all the different forms of a word. Instead we combine the word with the feature to find the correct morphological form. Numbers could in fact be processed by a special number grammar to avoid having to list all numbers in a lexicon. These tasks could of course also be solved using other neural networks.

The nodes in the output vector represents different syntactic categories, so we also get a surface syntactic structure directly output from the net, which could be used for stress information etc. to be input to a speech generator.

4 Results

The two networks were trained using backpropagation with momentum (learning rate 0.5), each training set consisting of 57 sentences was run for 600 cycles. For the first network the mean square error (MSE) was 0.08, for the second network 0.175. The number of hidden nodes in each network was 20, a higher or lower number of hidden nodes resulted in a higher MSE.

Using a threshold value of 0.8, the network could be said to have correctly learned the mapping from output to input activations.

Conclusion

Extensive experimentation is needed to define the proper typology and parameters for the neural networks. We especially need to experiment more with different learning methods. A future research topic will be to see what kinds of further subdivision of the tasks are needed. Elman suggests that simple recurrent networks could be capable of learning pronominal reference, and this would be an interesting extension of the system.

References

Fauconnier, G. 1985. *Mental Spaces: Aspects of Meaning construction in Natural Language.* Cambridge, Mass.: MIT Press.

Elman, J. E. 1990. *Finding structure in time.* Cognitive Science 14, 172-211.

Elman, J. E. 1991. *Distributed Representations, Simple Recurrent Networks, and Grammatical Structure.* Machine Learning 7, 195-225. Kluwer Academic Publishers, Boston.

Jordan, M. I. 1986. Serial Order: A parallel distributed processing approach (Tech. Rep. NO. 8604). San Diego: University of California, Institute for Cognitive Science).

Hovy & Lin 1999. *Automated Text Summarization in SUMMARIST.* In Mani & Maybury (eds.) Advances in Automated Text Summarization. MIT Press.

Kukich, K. 1987. *Where do phrases come from: some preliminary experiments in connectionist phrase generation.* In Natural Language Generation: New results in Artificial Intelligence, Psychology and Linguistics. Gerard Kempen (editor). Martinus Nijhoff Publishers, Dordrecht.

Ward, N. 1994. *A connectionist language generator.* Norwood, N.J. Ablex Pub. Corp.

TABLE 1		
Event elements	**Possible values**	**Number of units in output vector**
Theme	RABBIT, ARIADNE, REGINA, SINGLE_ROOM, DOUBLE_ROOM SINGLE_PRICE, DOUBLE_PRICE	7 + 2 units representing the feature number (possible values sing/plur)
Main_event	HAVE, COST	7
Complement	SINGLE_ROOM , DOUBLE_ROOM, ROOM, BEF	7 + 2 units representing the feature number (possible values sing/plur)
Subject_predicate	EXPENSIVE, REASONABLE, CHEAP, SMALL, BIG	7+ 2 units representing the feature number (possible values sing/plur)
Modifier	EXPENSIVE, REASONABLE, CHEAP, SMALL, BIG Numerical values (e.g. 20, 4000)	7 units

TABLE 2										
INPUT TO NETWORK 1										
	Name		**Service_domain**	**#single_rooms**	**#double_rooms**	**Price_single_room**	**Price_double_room**	**Location**		**Categorizer**
I1	1	1	1	20	0	0	0	0	0	0
I2	1	1	1	1	1	0	0	0	0	1
I3	0	0	1	0	0	4000	0	0	0	0
I4	0	0	1	0	0	0	7000	0	0	0
I4'	0	0	1	0	0	4000	0	0	0	0
I5	1	1	1	0	0	4000	7000	0	0	1
I6	0	1	1	0	0	0	0	0	1	1

TABLE 3						
OUTPUT OF NETWORK 1						
	Theme	**Main_event**	**Modifier**	**Complement**	**Subject_predicate**	**Location**
O1	000111101 (REGINA, sing.)	0001010 (HAVE)	20	001000110(SINGLE_ROOM, plur.)		
O2	000111101 (REGINA, def, sing.)		0010010 (SMALL)		000101101 (HOTEL, sing.)	
O3	0010001 (SINGLE_ROOM, sing.)	0000110 (COST)	4000	000001010 (BEF, plur)		
O4	000011001 (DOUBLE_ROOM, sing.)	0000110 (COST)	7000	000001010 (BEF, plur.)		
O4'	001000101 (SINGLE_ROOM, sing.)	0000110 (COST.)	4000	000001010 (BEF, plur)		
O5	000111101 (REGINA, sing.)				000100000 (EXPENSIVE, sing.)	
O6	000000101 (ARIADNE, sing.)		0000100 (CHEAP)	000101101 (HOTEL, sing.)		0000101 (CITY CENTER, sing)