A Word Hypothesis Lattice Corpus - a benchmark for linguistic constraint models

D T Modd - csxdtm@scs.leeds.ac.uk E S Atwell - eric@scs.leeds.ac.uk

March 1994

1 Introduction

In response to an input sentence, a typical recognition system (be it speech or handwriting) will build and output a word-hypothesis recognition lattice. That is a sequence of word candidate sets, where each word position is composed of a number of candidate words proposed by the recogniser. For example, on 'hearing' the sentence "Stephen left school last year", an English speech recognition system might produce the following lattice:

Stephen stiffens
left lift loft
school scowl scull
lest last lust least
yearn your year

To disambiguate such a lattice, a standard technique is to use a language model to constrain the possible choices, so that the chosen sequence of words is the most linguistically plausible (See, for example, Jelinek 90, Atwell et al 93, Rose and Evitt 92, Keenan 93, also other papers in this Proceedings). Unfortunately, no standard method exists for evaluating and comparing the relative effectiveness of the various linguistic constraint models available. It is believed that a benchmark lattice will act as a testbed for this. However, different recognisers produce different kinds of word hypotheses (e.g. Speech recognisers produce phonetically-similar candidates, whereas handwriting recognisers produce orthographically-similar candidates). Furthermore, representation or storage formats for lattices differ widely; also, candidates may have 'scores', 'likelihoods' or 'ranks' associated with them, and again there is a wide range of practices in the use and representation of such scores.

At Leeds we are attempting to collate samples of lattices from a range of recogniser sources. The collected lattices will form a **Lattice Corpus** , to be used as a standard benchmark for evaluation of large-vocabulary linguistic constraint models for speech and handwriting recognition. As the lattices are being acquired from many different sources, a standard representation format has been devised, to which all the collected lattices are being converted. The format has been designed so as to satisfy the following criteria: simplicity; clarity; and completeness.

The overall success of the project will rely to some extent on the willingness (and ability) of researchers in the field of speech and handwriting recognition to supply lattices. However a Lattice Corpus may be of use to the research community even if it contains contributions

from only a limited selection of sources as we have algorithms for generating artificial lattices by "ambiguating" text samples; and the standard generalised format and 'prototype' Corpus will constitute a firm foundation for future expansion.

2 The initial request

A letter requesting lattices was posted on relevant e-mail bulletin boards worldwide and on the Internet newsgroup comp.speech.

The initial response to the request letter was quite encouraging, indicating significant interest from potential users of such a linguistic resource; however the majority of researchers who showed interest in the project could not supply lattices immediately, for a range of reasons including copyright or commercial confidentiality restrictions.

3 The lattices

We started out with three lattices that had already been acquired. Two of these were handwriting recognition lattices - sent to us by Frank Keenan (OUP) - produced from a common input source; although the second lattice was much more sophisticated than the first. The third was an O.C.R. lattice provided by Tony Rose (Nottingham Trent).

Short extracts from these lattices can be seen below:

```
I

began

my

career
cart

in

commodity

broiling
broking

as

a}
el
2}
a
```

The first lattice from Keenan; each word position is separated by a blank line.

```
#1: *I*
#1: *began*
#1: *my*
#1: *career*
#1: *in*
#1: *commodity*
#1: *broking*
#1: *as*
#1: a} *a* #2: 2}
#1: *secretary*
#1: *with*
```

```
#1: *a*
#1: *brokerage*
#1: *company* Company
#1: Or #2: met #3: *then* #4: that #5: men #6: Of #7: diet #8: fit #9: the} #10: me}
#1: *became* Became #2: because #3: flame frame
#1: *a*
#1: *Personal* mortal
#1: *Assistant*
#1: *to*
#1: *four*
#1: *brokers.* brokers, brokers: brokers; brokers
```

The second lattice from Keenan; each candidate word has an associated numerical weight; and correct candidates are surrounded by asterisks.

```
he 183 231 18 112

may 191 222 17 23617

push 183 100 11 27474
rush 163 48 1 28724

for 198 232 2 96

increases 200 205 6 21482

in 147 232 2 126
is 105 232 21 133

money 200 182 1 24188

wages 238 97 6 33620
```

O.C.R. lattice in the format: word, recogniser-score, frequency, grammar-code, semantic-code.

Since then we have had lattices sent to us by further researchers outside Leeds including Stefan Besling (Philips, Germany), Russell Collingham (Durham) and Peter Wyard (BTRL).

4 Designing a standard format

4.1 Format requirements

We decided that the lattices should have the following format:

```
*Stephen* P1 P2 P3|stiffens P1 P2 P3|
*left* P1 P2 P3|lift P1 P2 P3|loft P1 P2 P3|
*school* P1 P2 P3|scowl P1 P2 P3|scull P1 P2 P3|
*last* P1 P2 P3|lest P1 P2 P3|lust P1 P2 P3|least P1 P2 P3|
*year* P1 P2 P3|yearn P1 P2 P3|your P1 P2 P3|
```

Each line contains one or more word candidates and represents a lattice-word position. Each candidate has a number of optional, associated probabilities. These represent the certainty that the candidate word is correct and are based on data provided in the original lattices, such as recogniser score, linguistic score, etc. Ideally we would like to have three probabilities associated with each candidate word: P1 - a probability calculated from the recogniser score; P2 - a probability calculated from the linguistic constraint model score; P3 - an overall probability calculated from P1 and P2. The sum of related probabilities on one line should equal one.

A delimiter is used to isolate word candidates and their associated scores. This also serves to make candidate extraction and manipulation procedures - using automated sequential searching techniques - easier to implement.

In order to identify the input sentence, correct candidates are specified. This is done by surrounding the correct candidate word with asterisks. Also, we felt that the correct candidate should appear at the start of the candidate set, although this is more an aesthetic requirement than a technical one.

In order to specify the strict syntax of the Corpus an EBNF grammar was constructed.

4.2 Definition of EBNF

An Extended Backus-Naur Form (EBNF) specification of the syntax of a language consists of a collection of rules or productions collectively called a "grammar" that describe the formation of sentences in the language. EBNF is widely used in the specification of programming language syntax, e.g. [Jensen and Wirth - Pascal User Manual and Report].

Each production consists of a non-terminal symbol and an EBNF expression separated by an equal sign and terminated with a period. The non-terminal symbol is a "meta-identifier" (a syntactic constant denoted by an English word), and the EBNF expression is its definition. EBNF cannot capture non-context-free constraints, so an EBNF grammar may need to be augmented with notes on semantics.

The EBNF expression is composed of zero or more terminal symbols, non-terminal symbols, and other metasymbols summarised in the table below:

```
Metasymbol      Meaning

=               is defined to be
|               alternatively
.               end of production
[X]             0 or 1 instance of X
{X}             0 or more instances of X
(X | Y)         a grouping: either X or Y
"XYZ"           the terminal symbol XYZ
Metaidentifier  the non-terminal symbol Metaidentifier
```

4.3 Format for an isolated word recognition lattice

The following is an EBNF grammar defining the syntax of the standard format for an isolated word recognition lattice, with accompanying semantic notes:

```
CandidateSet = [CorrectCandidate "|"] { Candidate "|" } EndOfLine .
Candidate = CharacterString { Probability } .
CharacterString = StringElement {StringElement} .
CorrectCandidate = "*" CharacterString "*" { Probability } .
Digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
EndOfLine = "\n" .
Footer = "-EndOfSentence-" EndOfLine .
```

```
Header = "-StartOfSentence-" EndOfLine .
Lattice = Header CandidateSet { CandidateSet } Footer.
LatticeCorpus = Lattice { Lattice } .
Probability = " 0." Digit { Digit } | " 1.0" .
StringElement = "**" | AnyCharacterExeptAsterisk .
```

NOTES:
1. The start symbol is LatticeCorpus.
2. A correct candidate is enclosed in asterisks (*); any other occurrence of * is coded as **.
3. AnyCharacterExceptAsterisk implies that any single character may act as the terminal symbol except "*".
4. Probability represents the probability that the candidate word is the input word according to weights such as acoustic score, linguistic score, etc. The sum of the related probabilities on one line should always be 1.

4.4 Overlapping candidates

The format described above assumes that there will be no overlapping candidates in the lattices, although this is obviously not always the case. In continuous speech recognition there are no distinct boundaries between input words and candidate words tend to overlap considerably; so a different format is therefore required which offers more information.

4.5 Format for a continuous speech recognition lattice

The format for continuous speech recognition has the additional information of a start time and end time for each candidate word, usually with the input starting at time 0 or 1. The start and end times can then be used to represent nodes in a Directed Acyclic Graph (DAG) in which the arcs represent the candidate words and their positions in time.

A correctly identified sentence will therefore be represented by an unbroken route through the DAG from the first node to the last.

The following is an EBNF grammar defining the syntax of the standard format for a continuous speech recognition lattice, with accompanying semantic notes:

```
Candidate = StartPoint EndPoint CandidateString { Probability } EndOfLine .
CandidateString = "*" CharacterString "*" | CharacterString .
CharacterString = StringElement {StringElement} .
Digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
EndOfLine = "\n" .
EndPoint = Digit { Digit } " " .
Footer = "-EndOfSentence-" EndOfLine .
Header = "-StartOfSentence-" EndOfLine .
Lattice = Header Candidate { Candidate } Footer.
LatticeCorpus = Lattice { Lattice } .
Probability = " 0." Digit { Digit } | " 1.0" .
StartPoint = Digit { Digit } " " .
StringElement = "**" | AnyCharacterExeptAsterisk .
```

NOTES:
1. The start symbol is LatticeCorpus.
2. CharacterString surrounded by asterisks (*) represents a correct candidate; any other occurrence of * is coded as **.
3. AnyCharacterExceptAsterisk implies that any single character may act as the terminal symbol except "*".
4. StartPoint and EndPoint represent nodes in a D.A.G.
5. Probability represents the probability that the candidate word is the input word according to weights such as acoustic score, linguistic score, etc.

5 Conversion program

A program has been written to convert isolated word recognition lattices into continuous lattices, wherein words in the same line in the former acquire the same start and end times in the latter.

Other programs for converting acquired lattices into lattices in the standard format are currently being developed.

These programs are all written in Pascal; this may not be the most suitable language for handling string manipulation but is highly portable, making the software more accessible to potential external users.

6 Use of final corpus

The final Corpus will be made available to Internet users and will be mailed directly to those researchers who have requested a copy.

Note that we have decided to 'keep' any linguistic scores with candidates rather than throw this information away: our EBNF grammars allow for more than one probability with a candidate.

Where lattices have been supplied with linguistic scores attached to candidate words, evaluation of the particular linguistic constraint model used is straightforward and the results immediately self-evident. In general, however, we do not know exactly what linguistic models were used in arriving at these scores, so they are best ignored. Users may wish to compare the success rate of their own linguistic constraints model against those of the lattices' originators.

7 Conclusions

Although there is a great amount of interest in this subject area, and in this particular project, it seems that there is not a lot of 'public domain' material available amongst the wider research community involved in the development of speech and handwriting recognition; where there is material researchers seem reluctant to share it.

We hope The results of this particular project are widely appreciated, and that more speech and handwriting researchers are spurred into donating samples to the public domain for the wider benefit of the whole Speech And Language Technology community

8 Acknowledgements

People who have contributed to this project so far are:

- Simon Arnfield - sca@scs.leeds.ac.uk
- Stefan Besling - besling@pfa.philips.de

- Russell Collingham - R.J.Collingham@durham.ac.uk
- Ido Dagan - dagan@research.att.com
- George Demetriou - george@scs.leeds.ac.uk
- Jason Eisner - jeisner@gradient.cis.upenn.edu
- Frank Keenan - fgk@oup.co.uk
- Kate Morton - kate@essex.ac.uk
- Fernando Pereira - pereira@research.att.com
- Tony Rose - tgr@doc.ntu.ac.uk
- Kripa Sundar - kripa@cs.buffalo.edu
- Michael Turner - turner@remarque.berkeley.edu
- Theo Vosse - vosse@ruls41.leidenuniv.nl
- Peter Wyard - wyard_p_j@bt-web.bt.co.uk

The authors gratefully acknowledge financial support from Kirklees MC, and UK HEFCs.

9 References

- Atwell, Eric, Simon Arnfield, George Demetriou, Steve Hanlon, John Hughes, Uwe Jost, Rob Pocock, Clive Souter and Joerg Ueberla. 1993. Multi-level Disambiguation Grammar Inferred from English Corpus, Treebank and Dictionary. In Simon Lucas ed. Grammatical Inference: theory, applications and alternatives. Colloquium Digest, Institution of Electrical Engineers.
- Jelinek, Fred. 1990. Self-organized language modeling for speech recognition. In Alex Waibel and Kai-Fu Lee (eds). Readings in Speech Recognition: 450-506. Morgan Kaufmann.
- Keenan, Francis. 1993. Large vocabulary syntactic analysis for text recognition. Unpublished PhD thesis, Dept of Computing, Nottingham Trent University.
- Rose, Tony, and Lindsay Evitt. 1992. A large vocabulary semantic analyser for handwriting recognition. AISB Quarterly. 80: 34-39.
- Jensen, Kathleen, and Niklaus Wirth. Pascal User Manual and Report: Third Edition. Springer Verlag.