Lexical Access via Phoneme to Grapheme conversion

[1]**Michael Fridkin,** [1]**David L. Dowe and** [2]**Simon Musgrave**
[1]Clayton School of I.T
[2]School of Languages, Cultures and Linguistics
Monash University
VIC 3800, Australia
mfri7@student.monash.edu.au
david.dowe@infotech.monash.edu.au
simon.musgrave@arts.monash.edu.au

Abstract

The Lexical Access (LA) problem in Computer Science aims to match a phoneme sequence produced by the user to a correctly spelled word in a lexicon, with minimal human intervention and in a short amount of time. Lexical Access is useful in the case where the user knows the spoken form of a word but cannot guess its written form or where the users best guess is inappropriate for look-up in a standard dictionary or by traditional computer spellcheckers. Previous approaches to this problem have attempted to match user-generated phoneme sequences to phoneme sequences in dictionary entries and then output the stored spelling for the sequences. Our approach includes a phoneme-to-grapheme conversion step followed by spelling-to-spelling alignment. This ensures that more distortion to the data can be handled as well as reducing error rates and also allows us to combine the Lexical Access problem with a related task in Natural Language Processing, that of spelling Out of Vocabulary (OOV) items. We call this combination the Hybrid Lexical Access (HLA) Problem. The performance of our system on this task is assessed by the percentage of correct word matches in the output after edit distance is performed to find the closest match to a stored item.

1 Introduction

In languages with irregular spelling systems, such as English, most speakers have trouble spelling at least some words. To aid spelling there are orthographic dictionaries (which index words by their sequence of letters) and phonemic dictionaries (which index words by their sequence of phonemes). We will be looking at how, given a phoneme sequence, phonemic dictionaries can be used by computer systems to look up single words and if the word is not in the lexicon then how computer systems can best give a suggestion to its true spelling.

In orthographic dictionaries, finding the correct word depended on being able to spell the first few letters correctly. Spellcheckers have improved on this method but still have two drawbacks. Firstly, they require a candidate spelling before they can begin matching and this can prove difficult in some situations; for example, an unstressed vowel sound (normally schwa in English) can be spelled with many different graphemes. Secondly, spell-checkers have difficultly matching multiple graphemes which can correspond to the same phoneme. For example, in the word Aachen, the candidates Acen, Aken and Acken would not map to the correct spelling using standard spellcheckers. Few speakers of English would know that /k/ is spelled as *ch* in some circumstances such as the one in the example. Computer systems can supplement the user's linguistic knowledge in these situations.

In Computer Science the *Lexical Access Problem* involves matching a sequence of phonemes to a sequence of words. It is assumed that each phonemic sequence corresponds to some word in the lexicon and it is the job of the computer system to ascertain what the intended word is. We wish to do a similar task but for single words and

without the assumption that the word is in the dataset. If the word is not in the dataset then the problem turns into that of spelling Out of Vocabulary (OOV) items. Obviously the user can not specify if the word is or is not in the dataset and the system must make that judgement itself. This extension of the Lexical Access Problem to include OOV items we call the Hybrid Lexical Access (HLA) Problem. Besides improving look-up of spelling for unfamiliar and unknown words, both aspects of this hybrid problem are also relevant as part of automatic speech recognition systems. The focus of this article is to compare and contrast the Lexical Access, Spelling OOV items and HLA problems and their applications to aid people with spelling.

2 Background

The dataset for the HLA problem is one or more phonemic dictionaries (the problem is restricted to single words so continuous speech data is irrelevant). Since dictionaries are transcribed by expert linguists or by automatically utilizing linguistic rules made by linguists and users are not expected to have the same training, there is no guarantee that both will agree on a correct pronunciation. The system must assume that most times there will be disagreement. The main role of the system is to standardize input from users to match, ideally, a single dictionary entry.

The system is given 1 attempt to produce the correct match for the input during testing but in practice it is allowed to output as many words as necessary (to cover the case of multiple homophones - i.e. to, two and too). We need to measure the system's accuracy by how accurately it performs across a wide range of words, to prevent bias towards one class of words, such as short words or common words or words with silent letters.

An example of a flaw in the system is if a dataset has silent letters, such as "w" in wrong, aligned to a special null or empty phoneme resulting in silent letters being second-class citizens and likewise the words containing the silent letters. If the "w" in wrong and "p" in psychology are treated as the same phoneme and there are 25 graphemes which may correspond to this phoneme in total then each will have a base probability of roughly $\frac{1}{25} = 4\%$. Such a system will have a high accuracy because it will give the correct output for the other phonemes. The output overall will be very close to being correct. The problem is that the bias will be directed towards words without silent letters. Those words that do not belong to this class will be significantly difficult for the system to find, roughly 25 attempts on average to find the "w" in wrong, which is not practically feasible.

Our system first does phoneme-to-grapheme conversion (PTGC) and then finds the best match from the dataset for that grapheme sequence (using the ASPELL spellchecker). If no match is found, then the generated grapheme sequence is the final output of the system. The metric which we use for measuring the success of the first stage of the system is a Average Conversion Percentage (ACP) at word level, corresponding to what (Marchand and Damper, 2000) refer to as "percent correct" and what (Rentzepopoulous and Kokkinakis, 1996) call "success rate of the conversion algorithm ... at the word (state sequence) level". The ACP is the percentage of entries converted correctly in the PTGC stage out of the total number of entries produced. No edit distance measure is included at this stage. During post-processing ASPELL matches the grapheme string output from the PTGC to a word in the dataset using the Minimum Edit Distance criterion which selects the grapheme representation for which the minimum number of insertions, deletions and substitutions is required (Levenstein, 1966). The success of the system at this stage is measured using ACP Average Conversion Percentage after Edit Distance (ACP-Ed), which again represents the percentage of correct matches out of the total number of outputs produced.

ACP-Ed was used to control training of the PTGC component and it has several advantages for this purpose. Firstly, it reduces any bias in the assessment process towards short words and common words. If Edit Distance is not taken into account, the performance of a system may be exaggerated if it handles such words well even if its output when dealing with longer words and uncommon words is very poor. Using ACP-Ed means that the system is guided towards produc-

ing good output for all input, even if very few inputs achieve perfect output. Secondly, ACP-Ed takes into account the size and distinctiveness of the available dataset and assesses not only how close the output of PTGC is to the correct graphemic representation, but also how close the output is to similar dictionary entries and whether sufficient information is available to discriminate between similar entries. For example, the phonemic input [a b d uh k t @r r z] ('abductors'), produces 'abducters' as output from PTGC. This has an Edit Distance of 1 from the correct graphemic representation (substitution of 'o' for 'e' is required at position 7), which is a good result. However, the dictionary also contains the entry 'abductees', which also has an Edit Distance of 1 from the PTGC output (substitution of 'e' for 'r' is required at position 8). Therefore, the PTGC output is not good enough to disambiguate between the two possible matches with grapheme strings from the dictionary and the result will count as a failure in the calculation of ACP-Ed.

When plotted on different axes, we see fluctuations between ACP and ACPed, indicating a non-monotonic relationship. Indeed, these fluctuations occur with some consistency across the two data-sets (UNISYN and *NETtalk*) used in our study and again when we varied the internal organisation of the PTGC process - namely, the format table encoding from sec. 5.4.

3 Previous Work

Pronunciation by analogy (Marchand and Damper, 2000) is a PTGC technique which matches substrings of the input phoneme sequence to substrings of a group of words with similar phoneme sequences in the dataset and uses their corresponding grapheme sequences to infer the output grapheme sequence. The system of that study used the *NETtalk* corpus of approximately 20,008 hand-aligned entries with stress and syllable boundaries to achieve 76.4% ACP.

NETtalk was produced by Terrence J. Sejnowski and intended for grapheme-to-phoneme conversion, it is setup using 50 phonemes and a null phoneme for silent letters. In practical situations the null phoneme, which gives cues to the system about locations of silent letters, is not given to the system. This makes this dataset an interesting area of investigation for phoneme-to-grapheme systems because accurately predicting how the null phoneme is spelled would greatly improve accuracy for silent letters. Roughly half of the *NETtalk* dataset contains the null phoneme. This is due to the fact that vowels can be monophthongs such as "e" in pet or dipthongs as in "a" in pay. The word pay is transcribed /pe-/ where the "e" stands for /eɪ/ and the y corresponds to the null phoneme. To make the dataset useful for training and testing PTGC systems, an auxiliary system needs to be created which takes as input a phonemic sequence and inserts the null phoneme in appropriate locations.

A run of the Snob (Wallace and Dowe, 2000) software on the *NETtalk* dataset revealed that roughly 70% of the time when the null phoneme was observed it was surrounded by predictable neighbours or one of several predictable neighbours. For example, the phoneme /eɪ/ represented in *NETtalk* by the letter "e", as a first vowel in the nucleus of a syllable receiving secondary stress in penultimate position, occurs 64 times in *NETtalk*. Out of those 64 times, 60 times the null phoneme is at the last position. To be even more precise, when /eɪ/ is spelled using the letter "a" and is the first vowel in the nucleus of a syllable receiving secondary stress in penultimate position it occurs 52 times in the dataset, all of which are followed by the null phoneme in last position. The predictability of the null phoneme can be the basis for a system for converting data in *NETtalk* format into one that is compatible with the general format of input to PTGC systems (which does not have the null phoneme).

The approach in (Thomas et al., 1997) to the Lexical Access problem was to match the input phonemic sequence to a phoneme sequence from the dataset and then output the corresponding grapheme sequence. Their dataset contained more than one realization of the same word so they used edit distance to match the input sequence to one of the realizations seen in the dataset from which the corresponding grapheme sequence was the output. The testing was done by inputting phoneme sequences and measuring how many input phonemes differ from a candidate stored lexicon sequence divided by the to-

VB - *Verbs*		
VBD	VBG	VBN
past tense	gerund	past participle
JJ - *Adjectives*		
JJR	JJS	
comparative	superlative	
NN - *Nouns*		
NNS	NNP	NNPS
plural	proper	proper plural

Table 1: Part of speech (POS) abbreviations

Orthography	a	b	o	a	r	d
NETtalk	x0	b	o1	-⟨	r	d
IPA	ˌɑ	ˈb	ɔː		ɹ	d
Unisyn	@	b	*oo	a		rd
IPA	ə	ˈb	ɔː			ɹd

Table 2: Comparison of dataset representations of the word *aboard* with different alignment strategies.

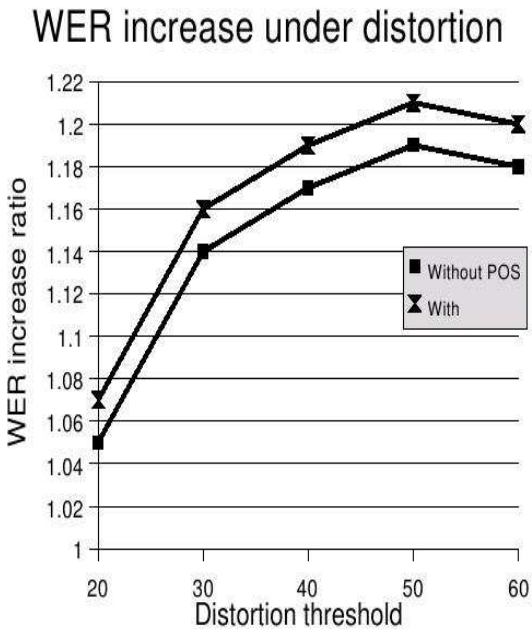

Figure 1: (Thomas et al., 1997) Results

tal number of phonemes, called the distortion rate. The distortion threshold is the maximum distortion rate for which the corresponding average Word Error Rate (WER) was calculated. With Part of speech (POS) tags in the input they achieved results of 24.53% WER under 10% distortion threshold and without POS tags, 31.29%. The degradation in performance of their system, with POS tagging and without, when increasing the distortion threshold can be seen in figure 1.

Two points of interest in figure 1 are that the last increase in distortion from 50% to 60% actually decreases the WER by 0.19% and that the model with POS tagging show a higher degradation under distortion than the one without POS tagging, which corresponds to our results for POS tagging (sec. 6). For example, when increasing the distortion rate from 10% to 20% the WER of the model with POS tagging increased by 1.62% whereas the model without POS tagging, with the same increase in rate of distortion increased by 1.56%. As a ratio of the original 10% WER, these come to 1.07% and 1.05% respectively as seen in figure 1, for 20% distortion. The 0.02% difference between the two models stays roughly constant as the distortion ratio increases.

4 Resources Used

There were two datasets used for training and testing our system, the UNISYN *Accent-Independent Keyword Lexicon* and the *NETtalk* corpus. The two datasets differed both in size and the levels of detail. The UNISYN dataset had roughly 119,356 entries, after entries with special symbols such as apostrophes were removed. Each entry contains orthography, POS tag, phonemic transcription, syllable boundaries, morpheme boundaries and typical word frequency as found in standard literature. However phoneme to grapheme alignment is not included and had to be done separately. The alignment strategy for this dataset, A_U, partitioned the dataset into phoneme and grapheme pairs using a hand-made table of intuitive correspondences.

The *NETtalk* corpus has for each entry an orthography, phonemic transcription and an indicator variable in the range 0 to 2 to say whether the word is irregular, foreign or common. The phonemes and graphemes are aligned one-to-one in a format suitable for grapheme-to-phoneme conversion. There is a null phoneme which corresponds only to silent letters and makes the correspondences more regular for the other phonemes.

The *NETtalk* alignment strategy we call A_N. A comparison of the results of the two models of alignment is in table 2.

To prepare the datasets for training and testing, lexical stress, POS tags and phonemic transcriptions were extracted from them and an alignment strategy was applied. For the *NETtalk* dataset the POS tags were matched to the UNISYN dataset where they had entries in common and the rest of the tags were automatically generated using *C&C Tools* (Clark and Curran, 2004).

For analyzing the datasets the Snob program was used, which does flat clustering using the Minimum Message Length (MML) criterion. When a dataset is run through Snob using an alignment model, A_N or A_U, the output is a set of classes all at the same depth for each pairing in the model. The classes are a hypothesis about the data that when applied to the data and transmitted with the data as a message has the shortest two-part message length (Wallace, 2005) (Wallace and Boulton, 1968) (Patrick, 1991). The premise is that the shortest message Snob finds will be a good explanation of the data.

5 Method

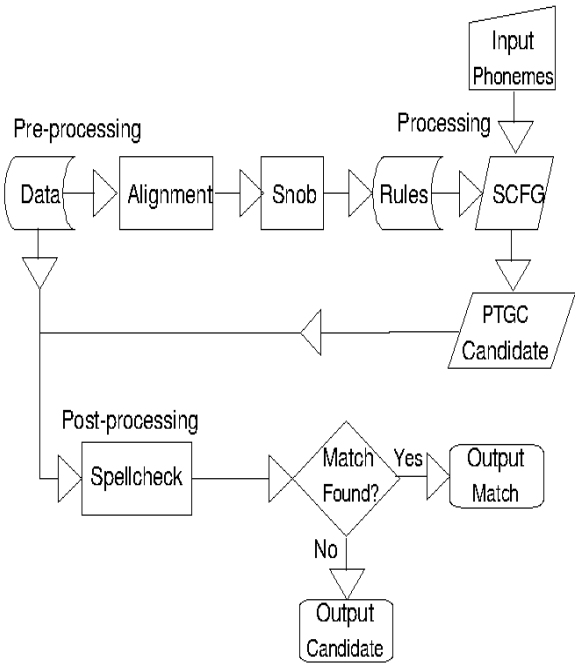

Figure 2: System flow

The system can be divided into 3 modules: (i) Pre-processing, represented in figure 2 as the steps from "Data" to "Rules", (ii) Processing, starting at the "SCFG" step and ending at the "PTGC Candidate" step, and (iii) Post-Processing, starting at the step "Spellcheck" and ending with either "Output Candidate" or "Output Match".

The pre-processing stage aligns the dataset by using an alignment model if the dataset is not already aligned and generates a list of classes produced by running Snob on the aligned dataset. The list of classes are passed to the parser to be converted into a Stochastic Context Free Grammar (SCFG).

The processing stage of the system is given the input in the form of a phoneme sequence and its POS tag, as well as a SCFG. Each phoneme in the sequence is processed sequentially and its output graphemes are concatenated to produce the candidate word.

The post-processing stage is given the candidate word with the dataset only containing the word spellings and it has to find the best match or fail and output the candidate word and part of speech tag it was given.

5.1 Pre-processing

The dataset usually consists of a series of entries of phoneme sequences and word spelling. In the phoneme sequence each phoneme is written out separately and the word spelling is written together as a single string. We want to look at the structure of the word to create models for understanding the relationship between the phoneme sequence and the grapheme sequence and this requires breaking up the word spelling into a grapheme sequence with each member of the grapheme sequence having one or more graphemes corresponding to one phoneme.

Phonemes with different levels of stress are considered as different, as well as phonemes that are the beginning of names. In total there are 657 pairs, 463 part of speech tag combinations (35 basic tags) - which are only used at the post-processing "Spellcheck" step - and 157 phonemes (including merged phonemes).

The dataset now consists of a series of pairs of phonemes and graphemes and each possible pairing is given a unique ID. A phoneme

can have multiple grapheme pairings so it spans over several ID's. We choose the ID's for each pairing carefully following the criteria that for each phoneme the ID's which correspond to that phoneme are contiguous, for example {39 k_c, 40 k_ck, 41 k_k}. ID's in each possible broad sound groups (such as monophthong vowel, diphthong vowel, stop or fricative) are also contiguous.

We can now view each ID in the list as a separate entity, distinct from the word in which it was originally found, surrounded by zero or more neighbours and having a position in the ID sequence. This list of independent ID's is used as input to the Snob program in order to generate classes which use patterns in the occurrence of neighbours to compress the dataset into groups whose combined size equals a fraction of the size of the original dataset.

Groups are chosen following two criteria. Firstly, the ID in any group is exactly the same or very close (for example, a group may consist of a vowel and a stressed form of the same vowel), and secondly the neighbours and position for that ID are as different as possible to all the other ID's which have the same phoneme (we do not need to worry about ID's that do not have the same phoneme because in the input we are given the phoneme). Each group is described by a Normal distribution for each of four neighbours (neighbour two spaces to the left to the neighbour two spaces to the right, and position of the ID). We end up with a distribution with the mean centered not at the most frequent ID but at the ID which has a value which is in between the most frequent ID's. These are our classes which from now on we refer to as rules.

5.2 Processing

The system parses an input phoneme sequence by looking at the environment (its neighbours and position) of each of the phonemes and looks up every relevant rule in the SCFG to find the most likely rule. All the rules we select have a distribution centered at one particular value for attribute 6 which is outputed by the parser when the rule is selected.

For example, for the pair "e ə" (meaning schwa is spelled with an "e" having ID 63 in our model (there are 1031 such pairs in a sample dataset of 10,000 words, as in "abhorr*e*nce". Snob will find a class with this identical spelling of schwa and the description of the attributes of this class will inform us when to use this spelling. This is the class which describes this pair:

Serial 517 Relab 0.019 Size 152.
Att 2 vals 152.
Mean 508.5465 S.D. 66.9917
Att 3 vals 152.
Mean 469.9997 S.D. 0.4144
Att 4 vals 152.
Mean 502.1750 S.D. 88.1242
Att 5 vals 152.
Mean 5.9335 S.D. 2.2393
Att 6 vals 152.
Mean 63.0000 S.D. 0.0400

Att1 is the model for the pair 2 spaces left, if there is none then the value is -1.0, likewise Att2 is the pair 1 space to the left, Att3 is the pair 1 space to the right, Att4 is the pair 3 spaces to the right and Att5 is the position of the ID for the current class. Most importantly att6 is the description of all the ID's in this class. The standard deviation of 0.04 means that all the ID's are exactly the same. Although there are 6 attributes Snob looked at, it chose the above 5 for this class and attribute 1 it deemed as insignificant for this class. Snob deems attributes as insignificant when their description is almost identical to the description of the population (for this attribute it is [Mean 246.7137, S.D. 210.8832]). Our current population is all possible pairings of schwa with any graphemes. To interpret this class we need to look at the range of the different Broad Sound Groups (BSG).

Start marker	-1	End marker	629
Monophthong	1 .. 215	Dipthong	216 .. 394
Tripthong	395 .. 404	Plosive	405 .. 454
Nasal	455 .. 494	Approximant	495 .. 546
Fricative	547 .. 574	Affricate	575 .. 618
Trill	619 .. 628		

This class says that we spell schwa as "e" when it is around position six preceded by a Plosive or Approximant and followed by "n" and then by a Plosive, Approximant or Fricative. These are the combinations "ment", "dence" and "tence". In our example the neighbours of the schwa are "ˈo r" on the left and "n s" on the right, that is Att1

is in the range 164 to 167, Att2 is 619 to 629, Att3 is 460 to 476 and Att4 is 559 to 574. All of the values lie without one standard deviation from where the class predicts except for Att2 which is two standard deviations away. This is still a high probability because 68.3% of the values lie within one standard deviation of the mean for the Normal distribution and 95.5% lie within two standard deviations.

In order to automatically find this rule we write all the classes we are interested in - that is all the classes that have an ID with the schwa phoneme - as production rules in the SCFG with attributes one to five on the right hand side and attribute six on the left. Next we give each class a weight which reflects the size of this class divided by the total size of the population. Then we give each input symbol a weight which reflects the probability of seeing this symbol when looking at the appropriate attribute for all the members of this class. Finally we choose the rule which has the highest probability of being parsed. A fragment of the SCFG for class 517 given the input "{ ˈo r ə n s 6 }" is:

1.0	Start	–>	St
0.15	e	–>	{ Att1 Att2 ə Att3 Att4 Att5 }
0.0017	Att1	–>	ˈo
0.0013	Att2	–>	r
0.0618	Att3	–>	n
0.0034	Att4	–>	s
0.1780	Att5	–>	6
0.13	St	–>	e

Non-terminals = {St, Att1, Att2, Att3, Att4, Att5}, start Symbol = {Start}, terminals = {e, ˈo, r, n, s, 6, {, }}. The weights for the St symbol represent the relative frequency of the ID in relation to the other ID's of this phoneme, the weight for the rule (surrounded by curly braces) is the size of the group it represents as a fraction of all the ID's of this phoneme and the weights of each attribute is calculated by sampling over the range of the input phoneme corresponding to the appropriate attribute. We sample from the Normal distribution which the attribute models, over the range of the input divided by the magnitude of the range to calculate the average probability of each ID which the input ranges over of belonging to this class. For example, attribute 1 is N(μ = 246.7137, σ = 210.8832) and the first input is "ˈo" with range 164 to 167 so the weight in the SCFG for this attribute given the current input is 0.0017.

Formally a SCFG is a pair (p, G) where: $p : P \rightarrow [0, 1]$ is a probability function assigning probabilities to rules such that $\sum_{i \in \Pi_A} p(A \rightarrow \alpha_i) = 1$ and Π_A is the set of all rules associated to A, (Benedí and Sánchez, 2007). A SCFG is consistent if the probability assigned to all sentences in the language of the grammar sums to 1.0.

The result of a Snob run (Wallace and Dowe, 2000) using the six attributes discussed at the start of the section was a set of classes, for each pairing of phoneme to graphemes, with each class having a model containing the mean, standard deviation and relative abundance. This produced a SCFG consisting of 35,000 classes over a dataset of 99,800 entries and 657 pairs, with an average of 53.27 classes needed to describe each ID.

Sys	Data	Size	ACP [a]	ACP[b]	ACP[c]
A_U[d]	UNI[e]	10K	16.53		84.74
A_N[f]	NET[g]	20K	31.27		91.00
A_N	NET[h]	20K	31.27		81.20
M.[i]	NET	19.5K		75.1	

[a] ACP when the system has no access to the lexicon
[b] ACP when the system has access to all but the target entry of the lexicon
[c] ACP when the system has full access to the lexicon, i.e. ACPed
[d] Our system using UNISYN alignment
[e] UNISYN dataset
[f] Our system using NETtalk alignment
[g] POS tags added to NETtalk
[h] NETtalk dataset
[i] (Marchand and Damper, 2000) under 0% distortion

Table 3: Comparison of Systems

From,To	[ə, æ]	[aɪ, æ]
Effect	-0.73%	-0.17%
Freq.	46.0%	6.78%

Table 4: Effects of Vowel Distortion

5.3 Post-processing

After a candidate is generated its most common part of speech tag is attached to it and an attempt is made to match the candidate to a lexicon word.

From,To	[d,ʃ]	[ʧ, ʃ]	[t, ʃ]
Effect	-3.11%	-1.21%	+2.39
Freq.	23.31%	3.05%	7.8%

Table 5: Effects of Consonant Distortian

From, To	NNP, NN	NNPS, NN	NNS, NN
Effect.	-1.85%	-0.76%	-3.7%
Freq.	14.4%	1.5%	18.0%

Table 6: POS distortion - Nouns

From, To	VBD, VB	VBG, VB	VBN, VB
Effect	-1.51%	-0.24%	-1.08%
Freq.	3.52%	6.7%	2.96%

Table 7: POS distortion - Verbs

From, To	JJR, JJ	JJS, JJ
Effect	-0.35%	-0.13%
Freq.	0.5%	0.71%

Table 8: POS distortion - Adjectives

If the candidate cannot be matched then the output is the candidate, however if the candidate can be matched to some lexicon word then that word is the output. In the testing stage if the output word has any homophones they are ignored however the end system will output all homophones with the output word.

5.4 Format Table Encoding

The Format table contains all the different pairings of phonemes and graphemes in the dataset. The table contains 657 entries for the phonemes and all the part of speech tag combinations. The table is used to process the data, which is in human-readable format into numeric form, for Snob to process. Different arrangements of the Format Table result in different SCFG's produced by Snob and different ACP and ACPed results.

6 Results and Discussion

There were three systems compared during the testing phase of the study, with our system tested on two different datasets under two different alignment models. Table 3 is a comparison of our system with that of (Marchand and Damper, 2000), with access to different amounts of the dataset. Comparing our results with (Thomas et al., 1997) was difficult because they used the WER metric which is calculated differently to our ACP and ACPed and their results are omitted from the table. Our system performs less well than the other systems when only PTGC is considered, but does better than either of the other systems when the post-processing is included. Only one system, that of (Marchand and Damper, 2000), was suitable for leave-one-out testing but looks to be the most promising solution to the HLA problem if the low ACP can be further lowered by the factor seen in our system when the system has full access to the dataset. The two models we used to test our system reveal that A_N gives the lowest ACP under all types of access to the dataset and that including POS tags increases the ACP from 81.2% to 91.0% for that dataset (table 3).

Two of the systems studied performed tests on the effects of distortion to their systems. Tables 4 to 8 are the results of running the A_N system under various distortions. The distortions were performed by changing all occurrences of a phoneme in the dataset to a different phoneme and running the system again without retraining it. The values in the tables are the penalty costs for each distortion in the corresponding row and column. For example, in table 4 column 1, row 1 the value of the penalty is -0.73%, corresponding to a fall in the ACPed of 0.73%. The "Freq." row reports the frequency of the column element in the dataset.

Tables 5 and 4 rely on the PTGC component of the system to recover from errors and tables 6, 7 and 8 rely on the spellchecking software to recover. Distortion to the input of the PTGC does not degrade performance as much as to the post-processor under most conditions of distortion unless the distortion results in a phoneme which is found completely different environments to the original. Table 5 actually records a bonus, suggesting that some phonemes have interchangeable environments. Tables 6, 7 and 8 demonstrate that during post-processing recovering from errors is difficult, with an average decrease in performance of 30% (where table 1 expands the abbreviations).

References

José-Miguel Benedí and Joan-Andreu Sánchez. 2007. Fast Stochastic Context-Free Parsing: A Stochastic version of the Valiant Algorithm. In *IbPRIA (1)*, pages 80–88.

Stephen Clark and James. R. Curran. 2004. Parsing the WSJ using CCG and Log-Linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 104–111.

A. Levenstein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics-Doklandy*, volume 10.

Yannick Marchand and Robert I. Damper. 2000. A multistrategy approach to improving pronunciation by analogy. *Association for Computational Linguistics*, 26(2):195–219.

Jon D. Patrick. 1991. Snob: A program for discriminating between classes. Technical Report CS 91/151, Dept Computer Science, Monash University, Melbourne, Australia.

Panagiotis A. Rentzepopoulous and George K. Kokkinakis. 1996. Efficient multilingual phoneme-to-grapheme conversion based on hmm. *Computational Linguistics*, 22(3):351–376.

Ian Thomas, Ingrid Zukerman, Jonathan Oliver, David Albrecht, and Bhavani Raskutti. 1997. Lexical access for speech understanding using minimum message length encoding. In *In UAI97 Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 464–471. Morgan Kaufmann.

Chris S. Wallace and David M. Boulton. 1968. An information measure for classification. *Computer Journal*, 11(2):185–194.

Chris S. Wallace and D. L. Dowe. 2000. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, January.

Chris S. Wallace. 2005. *Statistical and Inductive Inference by Minimum Message Length*. Springer, Berlin, Germany.