Pre-processing very noisy text

Alexander Clark
ISSCO / TIM
University of Geneva
UNI-MAIL, Boulevard du Pont-d'Arve,
CH-1211 Geneva 4,
Switzerland
Alex.Clark@issco.unige.ch

Existing techniques for tokenisation and sentence boundary identification are extremely accurate when the data is perfectly clean (Mikheev, 2002), and have been applied successfully to corpora of news feeds and other post-edited corpora. Informal written texts are readily available, and with the growth of other informal text modalities (IRC, ICQ, SMS etc.) are becoming an interesting alternative, perhaps better suited as a source for lexical resources and language models for studies of dialogue and spontaneous speech. However, the high degree of spelling errors, irregularities and idiosyncrasies in the use of punctuation, white space and capitalisation require specialised tools. In this paper we study the design and implementation of a tool for pre-processing and normalisation of noisy corpora. We argue that rather than having separate tools for tokenisation, segmentation and spelling correction organised in a pipeline, a unified tool is appropriate because of certain specific sorts of errors. We describe how a noisy channel model can be used at the character level to perform this. We describe how the sequence of tokens needs to be divided into various types depending on their characteristics, and also how the modelling of white-space needs to be conditioned on the type of the preceding and following tokens. We use trainable stochastic transducers to model typographical errors, and other orthographic changes and a variety of sequence models for white space and the different sorts of tokens. We discuss the training of the models and various efficiency issues related to the decoding algorithm, and illustrate this with examples from a 100 million word corpus of Usenet news.

1. Introduction

This paper addresses the problem of the pre-processing of noisy text. We concern ourselves just with English text, though we note that the same approach could be applied to many other languages. All texts are noisy to a greater or lesser extent, and even when post-edited will contain varying amounts of typographical errors. A useful assumption when processing such texts is that the errors are isolated and surrounded by clean context that can be used to correct the errors. Here we are interested in a more difficult problem -- processing texts where a high proportion of the words may have errors, and where indeed it may even be difficult to perform the tokenisation to identify the words. Our goal is to produce a tool which can process arbitrarily noisy text and even text which has been deliberately obfuscated for one reason or another - for example to avoid detection of discussions of an obscene or criminal nature. Here we discuss the design and implementation of such a tool, but we shall not present any extensive formal evaluations of this, partly due to current difficulties with obtaining suitable annotated data for testing. Our basic methodology which we shall describe in detail below is the standard noisy channel model employed widely in the fields of speech recognition (Jelinek 1997) and other areas of Natural Language Processing such as machine translation and spelling correction (Kernighan, Church and Gale 1990) an area quite close to our own. In brief, we consider that the text we study is being produced by an underlying stochastic process a Language Model, and that this data is then put through a noisy channel, a process that converts the sequence of symbols (abstract representations of words or word classes) produced by the model into a sequence of characters. In speech recognition the noisy channel is the acoustic model mapping the fixed set of words in the vocabulary of the system to a sequence of phones and then to acoustic features.

Here we have a radically different process: first, we must allow the system to have an unbounded vocabulary, and secondly the process by which the words are mapped to a sequence of characters is very different and needs to be decomposed carefully into a sequence of different levels, incorporating at the minimum the introduction of varying amounts of white space between the tokens, remembering of course that some tokens are written without intervening white space, various changes to the capitalisation of the words, including both the obligatory addition of capitalisation at the beginning of sentences and the optional use of capitalisation for emphasis and the removal of trailing periods from abbreviation occurring immediately before a period. In addition we have various types of typographical errors caused either by ignorance of the correct spelling or true typing errors due to lack of manual dexterity on the keyboard. We note that there are other ways of producing electronic text which include handwriting recognition systems on Personal Digital Assistants (PDAs), the output from speech recognition systems, and the use of OCR on printed documents. These will all have different patterns of error that we suspect could be handled using the system we present here with appropriate modifications. We do not however have any suitable sources of data to investigate this further at the moment.

2. Motivation

Studies involving very large corpora have often been based on the use of news feeds for obvious technical reasons exceptions are the Brown Corpus, the British National Corpus (BNC) and the forthcoming American National Corpus. News feed based corpora have advantages — they tend to be relatively homogeneous in register and in orthographic style (by which we mean the use of capitalisation, punctuation and white space), and this facilitates their processing by potential end users. Moreover, news feeds are an economically important source of information. Non-news feed sources of corpora have tended to come from several sources: one is of course transcriptions of naturally occurring speech, such as is represented in the BNC to a certain extent, or alternatively other sources of highly post-edited written text such as fiction. These corpora have their own problems — norms of informal spoken language tend to change quite rapidly, and fiction tends to have stylistically marked language that is quite uncommon in other registers. Informal written language is becoming more and more widespread. Usenet, a distributed news service that is still a thriving forum of often acrimonious debate, and other forms of electronic communication with text are growing in popularity, whether on the Internet (IRQ, ICQ) or on mobile phones (SMS). The study of this language is intrinsically interesting and deserves study itself. Additionally, it can be a very useful source for other studies of language. Our particular interest is in extracting adjacency pairs for dialogue processing using quoted sections from Usenet posts. We shall provide some illustrative examples from a corpus of 100 million words of Usenet posts that we have prepared over the past two years.

3. Pre-processing

It is now time to make precise the set of tasks that we wish to perform. We conceive of the processing of texts as a sequence of operations starting from some sort of computer file or stream, and ending with some more detailed and structured representation. We consider the very first step -- the parsing of the data format, be it raw text in some character encoding, or the parsing of some structured text format such as XML or the native file format of a commercial word processor to be done as a preliminary step by other components. The stages that we are concerned with in this paper take as input the sequence of characters (we use Unicode throughout) including white space characters and so on. We wish to produce from this a sequence of tokens together with certain simple analyses. In particular we wish to identify

- The boundaries of the tokens: that is to say the sequences of characters that form each token
- The boundaries between the sentences that form the texts
- The correct spelling of any words that have typographical errors in them
- Whether words have had their natural capitalisation altered at all.

First it is worth pointing out that the exact definition of a text token is a far from a clear matter. We take an agnostic view on this matter. There are several standard tokenisations that are used in English, such as the Penn treebank (Marcus et al. 1993) or the BNC tokenisation (Burnard 1995), use several different criteria. Amongst the differences between these tokenisation schemes are whether clitics are separated from the preceding word, whether auxiliaries with attached negative particles are detached (for example is isn't decomposed into two tokens is and n't) and whether complex prepositions are considered to be single tokens or not (as far as). In other languages there are similar questions to be addressed. In German uses noun-noun compounding extensively, and such compounds are written without internal white space. It is perfectly legitimate to consider the segmentation of these compounds as part of the tokenisation process.

We will now show a few simple examples that illustrate the range of errors that we want to account for, and will explain why such a seemingly simple task requires such as complex tool.

Table 1 Examples of noisy text

1	I did with my last one.....

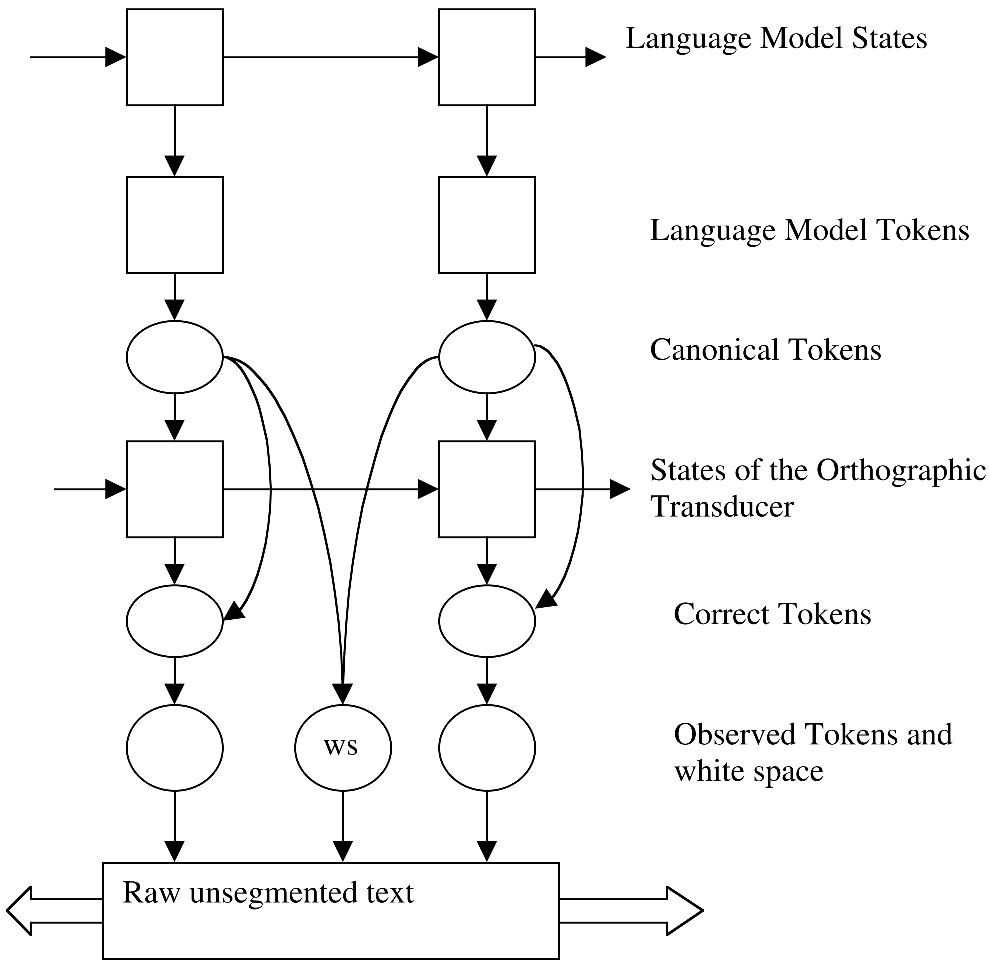
	She is now 3 1/2 and talks extreemly well for her age Yet at one she signed a number of words, I think the only thing it did was stop her getting frustrated when she wanted to communicate.. She could tell me if she wanted a hot drink or cold drink.....if she was hungry, tired, etc........having said that...my first 2 children didnt learn to sign BUt could still communicate in different ways..
2	Yep. The Despots NEED Street-Level Thugs to EmPower them in their Vile Political Manipulations.
3	E nergy by itself has no use in respect to our topic,howi understood it.
4	I really don’t know where to start on this. Dj’s have been losing thier gear to the government (local, state, federal, whatever) for years.
5	For the most part, I’d say yes-they are well-pigmented.

To give a concrete example of the output of the pre-processing we consider that given the second line of the first example in Table 1, the output produced should consist at the very least of the information that the text consists of a sentence missing a punctuation mark at the end, that the initial word She has been capitalised since it is in sentence-initial position and that the word extremely is a misspelling of extremely . In terms of the tokenisation, one could treat the string 3 1/2 as 1, 2 or 4 tokens depending on how finely one wishes to decompose the text.

It is worth pausing now to consider why this problem is difficult. The basic problem as in so many problems with language is ambiguity. The orthographic features that are used in writing text are used to mean different things in different places. Thus the use of capitalisation is used both to mark sentence-initial words, and also to mark proper nouns and adjectives (France, British). The period is used both to signify the removal of material from an abbreviation, and also to mark sentence boundaries. The dash or minus sign - is used both as a hyphen, as a minus sign, to signify various ranges as a parenthesis and so on. Some of these in theory should be represented by different characters (-,—,) and so on, but in the texts we are dealing with this will not happen. When we add to this the problem of interpreting noise, and correcting spelling mistakes, it is clear that the problem is non-trivial. Moreover, many of these problems are interlinked. First, identifying sentence boundaries requires resolving the ambiguity of capitalisation, and of periods. Secondly, spelling errors frequently introduce errors in the tokenisation. A very common example in English is the incorrect use of the apostrophe before plurals, and the confusion between its and it’s. On rare occasions we also have errors that introduce ore remove white space such as in Example 3 above. In addition compounds like law-abiding in English are often written with a hyphen or a space to mark the compound boundary. Example 5 above is a good example of the difficulties involved in resolving this. Moreover correcting the spelling errors requires a tokenisation in order to get the sentential context that can identify the words that might be misspelled. Thus we see that we have a tangled problem with these various processes interlocking in a rather complex way. We argue therefore that the appropriate solution is an integrated tool that can simultaneously perform the tokenisation, spelling correction, sentence boundary detection and capitalisation processing, and find an optimal or near-optimal solution, computing over all areas of ambiguity at the same time.

There are a number of further areas of processing that also need to be done. Notably amongst this is the normalisation of non standard words as discussed by Sproat et al. (2001). Though an interesting and important area we feel that this can be deferred to a later stage of processing using basically a standard HMM tagging framework as it does not exhibit the same pernicious integration with the other processes. Similarly part of speech tagging and shallow parsing, and word sense disambiguation can all be performed later without loss of precision. An initial area of processing we have considered is also one specific to this corpus and to corpora of emails: namely the task of segmentation into text and non-text blocks which requires the identification signatures, quoting and other problem areas. We assume that this has been performed prior to the processing described here.

Previous Approaches



We now describe briefly some previous approaches to this and related problems. We are not aware of any systems that integrate the processing to the extent we advocate here though the work of Mikheev (2002) is an interesting step in this direction. There are also standard techniques for spelling correction. Kernighan, Church and Gale (1990) is a seminal paper in this regard introducing the use of a noisy channel model to perform spelling correction. The idea here is to find the most likely sequence of words that could have given rise to the observed sequence of tokens. This involves two components: a language model that will predict the sequence of correct words and an error model that models the conditional probability of the observed string given the correct string. This work has been extended since notably by Brill and Moore (2000) who use a more sophisticated error model to get substantial improvements. Note however that they are primarily concerned with errors in post-edited text and thus their model is designed to account for ‘‘plausible’’ errors i.e. errors where the author might not notice the error or might believe it to be correct. Limitations of this work include the fact that generally they do not look for errors in words that are in the dictionary. This is a serious flaw since a large amount of errors do in fact result in correctly spelled errors. For example, from is often misspelt as form which is in the dictionary. Thus authors using simple spelling correction tools such as are often integrated into email clients and such like often will produce a high proportion of errors such as this which are overlooked by the software.

Tokenisation of Asian languages, such as Chinese, Japanese and Thai present particular problems for text processing since they are in general written without explicit indications of token boundaries in the form of white space. This has given rise to a series of algorithms using in general a simple Viterbi decoder to choose the optimal segmentation (Teahan et al. 2000). An approach that has some points of similarity with our own is presented in Gao et al. (2000). Though in English we do in general have white space this is not always used as is pointed out by Sproat et al (2001). In that paper they analyses a corpus of real estate advertisements which because of a cost per character are often written in a very compact way frequently removing white space. Among other algorithms they propose an algorithm using Finite-state automata and a Viterbi decoding algorithm. Again, there are points of similarity with the approach presented here.

Tokenisation in English has not been extensively studied *per se* largely because on clean text very accurate results can be obtained, using basic tools such as sed or Perl or some other regular expression based tools such as the Text Tokenisation Tool (Grover et al. 2000).

4. Outline of approach

We use a straightforward machine learning methodology using generative models and a noisy channel method. That is to say we produce a generative model for the sequence of characters in noisy text, that is based on a sequence of tokens separated by white space. Given a sequence of characters, we can then use a decoding algorithm to find the most likely sequence of tokens that would have generated the observed text; more formally, the sequence of tokens which has the highest probability given the observed unsegmented input. This is as previously discussed rather similar to the standard noisy channel model for spelling correction (Kernighan, Church and Gale 1990).

There are a number of alternative methods that could be used -- indeed any method for chunking or shallow parsing could be adapted to this task, just by converting letters to words. However these methods would fail to take account of the multi-word context required to disambiguate the sort of problems displayed by the examples above.

As previously described, our view is that the problem requires an integrated approach: namely, rather than treating tokenisation, sentence boundary detection and spelling correction as separate modules joined together in a pipeline, it is necessary for processing noisy text to integrate the two problems. Our approach is not here to take a system designed for clean text and gradually modify it to handle certain hard cases, but to design a system from the ground up to be able to process arbitrarily noisy text. In the next two sections we describe first the basic stochastic model that we use and its various independence assumptions and secondly the specific models and algorithms that we have chosen to implement.

We now describe the model on which we base our algorithm. Effectively here we are doing language modelling at the character level rather than at the word level, but incorporating an n-gram word model. We consider the sequence of characters as consisting of a sequence of tokens interspersed by white space which can be of zero length. This is a rather abstract view since we allow white space to occur inside tokens and also allow non-white space characters to occur inside the white space. We consider the sequence of tokens to be generated by some sort of stochastic process that we model by a language model.

A key element of our approach is that we model the process as having a sequence of levels of description for tokens, ranging from abstract to the observed strings, together with a set of models that govern the production of the more specific level from the more abstract. We will describe them briefly here and then present some examples that will clarify the definitions.

- The Observed string is the actual sequence of characters that we observed in the input that may have been corrupted by typographical errors.
- The Correct string is the sequence of characters that has been uncorrupted. This may not be correct in the sense of conforming to norms of written language.
- The Canonical string is a representation of the string abstracted from some orthographic features such as capitalisation.

Thus the simplest case is that all three levels are the same. If a word is misspelt, then the observed string will differ from the correct string, if the word has been capitalised or altered deliberately in some way, then the correct string will differ from the canonical string.

Orthographic Model

The canonical form of a word is just the form it is normally written in, in sentence internal position. The correct form is the form that it should be written in. There are two changes that can be made to the canonical form: first, the word can have its capitalisation altered, and secondly the trailing period can be removed (when the word occurs before a period or ellipsis). The role of the orthographic model is to handle the transduction from the canonical form to the correct form. There are a number of constraints on how this should be done — it must be deterministic given both the input (canonical) and output (correct) forms, and it should be correctly normalised. Since for technical reasons we want the program to operate without look-ahead, we need to allow some limited non-determinism.

Error models

We next allow the process to misspell or mistype normal words. We will use this at the moment to model two distinct processes typing errors and spelling errors. These two have rather different properties. True typing errors are caused when the typist intends to produce a particular sequence of key-strokes and fails; spelling errors are caused when someone genuinely believes that a word is spelt in a particular incorrect way. They differ in two respects: first typing errors generally consist of metathesis (the transposition of two adjacent letters), and substitution errors where an adjacent key is substituted for the correct, such as for example f being substituted for d . Spelling errors, on the other hand, often involve phonological proximity particularly with vowels being substituted for each other. The second difference is that typing errors are more independent stochastically. Spelling errors will often be repeated when the word concerned re-appears in a text written by the same author. In post-edited texts, we need also to distinguish a third process: accidental typographical errors that are not corrected during the editing process. In this case we will have a subset of typing errors that are superficially plausible. Glaring errors such as an insertion of an f before a d will be corrected, and more subtle errors will remain.

White space type

We condition the white space model on the preceding and following token. This seems to be the minimum necessary. The fact that periods and other punctuation marks are written immediately following the previous token, and the fact that currency symbols are written (generally) immediately preceding the quantities they modify seems to justify this. We are not aware of any longer distance effects where white space is a factor. Note also that white space following a sentence boundary often has one or more new lines, and that some punctuation marks are often followed by more than one space.

Discussion of overall model

The diagram in Figure 1 shows the overall structure of the model. The complexity of the model allows a certain amount of flexibility in how we treat particular phenomena. So for example some people do not use any capitalisation at all. This could be modelled either by the orthographic model, or by the error model. To model it using the orthographic model we add a state or states that represent the removal of all capitalisation, and add some transitions, and the appropriate transductions. Modelling it using the error model would involve having either a higher probability for substitution errors such as A being replaced by a , or modelling the errors as a composition of two transducers, one which models the changes in case, and the other which models the errors which change the characters further. Clearly the independence assumptions in this case favour the former approach, and accordingly we have used this.

There are however a number of situations where the model is insufficiently general. First, there are limitations to do with the decoding algorithm: in particular the model cannot correct words that are not in the word list of the fast match. Thus if one encountered a new word that ended in ollogy , even without having encountered the corresponding word ending in ology , orthotactic constraints would indicate that it is misspelled. It is technically rather difficult to model this process, and it is not clear how important it is in practice. Secondly, there are a broad range of errors where the errors are not limited to an individual token that cannot be processed by this model. In particular, typographical errors that span more than one token cannot be handled correctly. e.g. isn’t misspelt as ins’t, a fairly common error that occurs about 50 times in our corpus. This can be handled as two separate substitution errors, but not as the transposition. Next, there are some errors which involve the sequence of tokens themselves, such as tokens that are duplicated or omitted. Examples such as the following can be found easily

> Also I found the the ground here
> this year does not seem to hold water?

They could perhaps better be handled by a tagging/shallow parser component, just as the use of repetition deliberately very, very, very worrying is handled, or ellipis in general. Tmesis is also a phenomenon which is difficult to handle with this model.

> Fan-Fucking-Tastic. I know this film has its detractors. Too many people expecting another "Martin" or "Dawn Of The Dead".

Spelling mistakes inside composite tokens are also not treated properly. If the compound token is in the fast match then it will be handled properly, otherwise we would need a separate component for dealing with this. Since such a component is probably necessary anyway, we do not feel this is a serious problem, as it does not interfere with the tokenisation.

5. Specific Algorithms

We now can describe the algorithm based on this model more specifically. Our decision to treat all of these problems at the same time obviously leads to substantially increased complexity. We have therefore taken care to decompose the tool into various modules, with cleanly defined interfaces that can be altered separately without causing problems. The main modules of the system thus correspond basically to the various statistical models we have used and are as described in the following sections. This modularity has some negative aspects. First, there is a certain amount of overlap in the lexical resources used in identifying the classes. So for example clearly the type of a token as defined by the white space model and the type used by the orthographic model may have substantial overlap. This causes inefficiencies both in terms of time (the same classification may be performed by more than one component) and in terms of space (since word lists and so on may be loaded by more than one component).

The language model we use is a trigram language model with interpolated Kneser Ney smoothing.(Ney et al. 1994). This is a deterministic model with the states of the language model corresponding to bigrams of the language model tokens. Any other deterministic language model could also be used. Rather than having an out of vocabulary (OOV) token that we ignore, we have what is sometimes called a hierarchical language model (Galescu and Allen 2000) that incorporates separate models for the sequences of characters that make up unknown words. We use several different classes of OOV token here that can be used to model the unknown words more precisely. We will not give an exhaustive list of the classes here since to certain extent this is application dependent. We will merely outline the circumstances under which it might be desirable to have a separate type. The first types are of course just basic words which have a single invariant form. The vast majority of forms are like that. Next we have true out-of-vocabulary tokens representing words that we have not seen yet. In practice since we want to model the co-occurrence of these tokens with tokens we have already seen, we will also include a section of words that we have seen already but that are very rare. We will divide these into classes: at the very least separate classes for numbers and normal words, but optionally also separate classes for different part of speech tags for example. Technically we would wish the sets of strings in each class to be disjoint but this is not vital -- the only result would be a slightly deficient model. In addition there are two other situations in which it might be desirable to model the token as a set of strings. First we have variant spellings of particular words, whether due to dialect variations (colour/color) or register variations (tonight/tonite). Modelling these alternations reduces the amount of data required. Secondly we have tokens which have a variable length. The following tables will illustrate some of these. First, the use of exclamation marks and question marks is rather free, as can be seen in this table which shows the frequency of occurrence of strings of up to 10 symbols.

Table 1: Productive use of punctuation marks.

80724	!	220501	?
3682	!!	3675	??
3719	!!!	2857	???
1044	!!!!	738	????
639	!!!!!	291	?????
262	!!!!!!	113	??????
202	!!!!!!!	73	???????
106	!!!!!!!!	79	????????
101	!!!!!!!!!	15	?????????
70	!!!!!!!!!!	8	??????????

The next table illustrates the use of another word no in our corpus, together with a comparison with the frequency of occurrence in a million words of the Penn Treebank. As well as illustrating the necessity of a productive model for this word, it highlights the vast difference in the number of examples one can get from this sort of noisy corpus, compared to a small clean corpus, when it comes to an extremely common word such as no .

Table 2: Comparison of counts of variants of "no" in Penn Treebank and the Usenet corpus.

String	Penn Treebank count	Usenet count
no	936	404114
No	111	45429
NO	2	12831
noo	0	26
Noo	0	31
NOO	0	1
nooo	0	7
Nooo	0	2
NOO	0	4

Similarly the following table shows a small subset of the ways in which authors express laughter and surprise. This class has a large number of different types attested, and is a very good argument for showing that we need a model at the word list more sophisticated than a simple list of possibilities.

Table 3: A productive word class.

ha	7459
Ha	2617
HA	1887
haha	664
hahaha	249
hah	214
ha-ha	213
Hah	213
Hahaha	162
hahahahaha	152

Fast Match

The Fast Match is the component that we use to identify possible correct spellings of observed words. Given an observed string in the input, we normalise it by removing non-alphabetic characters and then retrieve from the fast match all words that are sufficiently close in terms of Levenshtein edit distance. In general, the word list of the fast match should be much larger than the vocabulary of the Language Model. We have experimented with various algorithms for this including Burkhard-Keller trees but have obtained best results by using a Prefix Tree Acceptor built on the reversed strings, where it is easy to compute the Levenshtein distance for all suffixes of a given string with all the words in the word list very efficiently.

Orthographic Transducer

The Orthographic model is responsible for the mapping between the canonical forms and the correct forms. There are basically two sorts of changes that are made. We model the mapping between canonical forms and correct forms as a fully-aligned non-deterministic stochastic finite-state transducer. There are slight problems with the normalisation of this model. There are basically two independent processes that need to be modelled one which is the use of capitalisation, that is conversion of upper and lower cases, and the other which is the removal of trailing periods from abbreviations before periods and ellipses. Neither of these processes is performed consistently, and thus we must model them as a stochastic process. We model them as two independent transducers and use the product of them which reduces the number of parameters and the complexity considerably. To give a concrete example the transducer models the transduction from the string of canonical tokens perhaps , but even if I did, that in no way means to the observed text which is Perhaps, but even if I did, that in NO WAY means . Given a particular input string of tokens, there are many possible output sequences, since emphasis can be applied in many different places. A problem with this mode of analysis is that because we assume that we lose the information that particular phrases are more often emphasized than others. Of course, if this turned out to be a problem, in practice we could add the emphasized tokens to the language model.

White Space Models

The white space model is not very critical. There are basically three or four different circumstances we want to distinguish. First, we have the normal situation where we have the white space between two normal words which is normally represented by a single white space character or a new-line character if it is at the end of a line. Secondly, we have the space between a word and a clitic which is invariably of length zero. Thirdly we have the space between currency symbols and the numbers they modify, which is normally zero but frequently has spaces. Finally, we have various classes that model the white space before and after and between various different classes of punctuation symbols. The specific models we use are character trigram models with boundary symbols, which seem to have enough structure to capture the limited length dependencies.

Error Model

We use trainable stateless stochastic transducers to model typographical errors. We bootstrapped some models using the data presented in Kernighan, Church and Gale (1990), but we found the distribution of errors in our data to be radically different from the results they report. We intend to experiment with more sophisticated error models, but at the moment we feel the additional computation cannot be justified. We focus rather on modelling the particular classes of errors that are critical to tokenisation, such as incorrect use of apostrophes.

Viterbi Decoder

The decoder is the component which determines the most likely sequence of language model tokens given just the observed strings. We use a Viterbi decoder with a beam search. For each character in the input we store the hypotheses that terminate at that point together with the log probabilities. We only store the best hypothesis that ends in a given language model state and state of the orthographic transducer. In addition in order to reduce the search space we only consider further those hypotheses that are sufficiently close to the best hypothesis. This can take the form either of considering only the n-best hypotheses, or considering all those hypotheses whose log probabilities lie within a certain amount of the best hypothesis. This is necessary since otherwise we would have to consider a number of hypotheses at worst equal to the square of the vocabulary size. In addition we use some *ad hoc* heuristics to prune the set of tokens we consider at each character, based on the unigram probability and the error model probability. The precise form of the algorithm is rather complicated since we need to operate the orthographic model in both directions, producing candidate tokens for words in the fast match, and also reversing the process and reconstructing the canonical form for new words. We shall not describe it here.

6. Discussion

We have presented a tool for performing a closely related set of low-level processing tasks on arbitrarily noisy text. We argue that the intertwining of these different models makes such an integrated tool necessary. Part of the motivation for this work has been to acquire lexical resources from less formal registers of English that could then be used in further processing and in providing more appropriate language models for conversational speech that newspaper or newswire derived ones. A second motivation is to provide very robust front end for written natural language dialogue; accordingly we have implemented this tool in such a way that it can run asynchronously with the input stream in a multi-threaded environment. At the time of writing the tool is fully implemented in Java, and we are engaged in tuning the models, and performing various boot-strapping operations. The modularity of the system allows us to run it in various different modes, for example assuming that the text has no spelling errors, or has all tokens separated by white space. We are using the Penn tokenisation on a large corpus of English Usenet posts.

In terms of future work, our most pressing concern is to perform a proper evaluation. Since it is very time consuming to manually annotate data with tokenisations, currently we use a crude tokeniser to produce a first draft that we then manually correct. This will obviously tend to under-report errors significantly. A further problem is that the gold standard is not always perfectly clear. Frequently there can be different possible analyses that are fairly plausible. A check on inter-annotator agreement seems to be necessary. Additionally, the data we use is very mixed. Much of our Usenet corpus is very clean and highly edited. Frequently people post excerpts from press releases or copyrighted news material. We are mostly interested in the noisiest segments of this data, where the additional complexity of our approach is justified.

Currently we use single models for many of the components. An improvement might be to use mixture models which could track different styles of typographical errors. A more challenging problem is to separate intentional errors due probably to ignorance of the correct spelling, from accidental typographical errors, for improved batch processing.

There are a number of other areas in which people use rather non-standard orthography. The use of asterisks or underscores as emphasis _really_ or *very* is quite common. Sentence boundaries are not always marked with a specific punctuation mark, but sometimes just by white space (in particular new lines). Line length in this case is an important clue.

Acknowledgments

This work has been supported by the Multimodal Dialogue Processing IP (issco-www.unige.ch/projects/im2/mdm) of the IM2 project (www.im2.ch).

References

Brill E, Moore, R C. 2000 An improved error model for noisy channel spelling correction. *Proceedings of ACL 2000*.
Burnard L 1995 *Users Reference Guide for the British National Corpus*.
Galescu L, Allen, J.2000 Evaluating hierarchical hybrid language models. *Proceedings of ICSLP 2000*, Beijing, China, October.
Grover C, Matheson C, Mikheev A, Moens M 2000 .LT TTT a flexible tokenisation tool. *Proceedings of LREC 2000*.
Gao J, Wang H, Li M, Lee K-F 2000 A unified approach to statistical language modelling for Chinese. *Proceedings of ICASSP*, Istanbul, Turkey,.
Jelinek F.1997 *Statistical methods for speech recognition* MIT Press.
Kernighan M, Church K, Gale W 1990 A spelling correction program based on a noisy channel model. *Proceedings of COLING 1990*, pp 205-210.
Mikheev A 2002. Periods, capitalized words, etc. Computational Linguistics, 28(3):289—318.
Marcus M, Santorini B, Marcinkiewicz M 1993 Building a large annotated corpus of English: the Penn Treebank *Computational Linguistics*, 19.
Ney H, Essen U, Kneser R 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1—38.
Paul D, Baker J 1997. The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the DARPA SLS Workshop*, February.
Sproat R, Black A, Chen S, Kumar S, Ostendorf M, Richards C.2001Normalization of non-standard words. *Computer Speech and Language*, 15(3):287—333.
Teahan W, Wen Y, McNab R, Witten I 2000 A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, 26(3):375—393.