Recognition Assistance

Treating Errors in Texts Acquired from Various Recognition Processes

Gábor PRÓSZÉKY
MorphoLogic
Késmárki u. 8
1118 Budapest, Hungary
proszeky@morphologic.hu

Mátyás NASZÓDI
MorphoLogic
Késmárki u. 8
1118 Budapest, Hungary
naszodim@morphologic.hu

Balázs KIS
MorphoLogic
Késmárki u. 8
1118 Budapest, Hungary
kis@morphologic.hu

Abstract

Texts acquired from recognition sources—continuous speech/handwriting recognition and OCR—generally have three types of errors regardless of the characteristics of the source in particular. The output of the recognition process may be (1) poorly segmented or not segmented at all; (2) containing underspecified symbols (where the recognition process can only indicate that the symbol belongs to a specific group), e.g. shape codes; (3) containing incorrectly identified symbols. The project presented in this paper addresses these errors by developing of a unified linguistic framework called the *MorphoLogic Recognition Assistant* that provides feedback and corrections for various recognition processes. The framework uses customized morpho-syntactic and syntactic analysis where the lexicons and their alphabets correspond to the symbol set acquired from the recognition process. The successful framework must provide three services: (1) proper disambiguated segmentation, (2) disambiguation for underspecified symbols, (3) correction for incorrectly recognized symbols. The paper outlines the methods of morpho-syntactic and syntactic post-processing currently in use.

Introduction

Recognition processes produce a sequence of discrete symbols that usually do not entirely correspond to characters of printed text. Further on, we refer to this sequence as an *input sequence*.[1] A unified linguistic framework must perform a transformation where (1) the symbols from the recognition process are converted into characters of written text, and (2) the correlation between the original analog source and the result is the closest possible. A post-processing framework must not simply perform a symbol-to-symbol conversion. A direct conversion is either impossible (phonetic symbols of any kind do not directly correspond to printed characters) or insufficient (source symbols are underspecified or incorrectly recognized). Morpho-lexical and syntactic models can help this process as they recognize elements of the language, extracting meaningful passages from the input sequence.

Lexical databases with fully inflected forms are fairly standard for speech recognition, mainly where a small closed vocabulary is used, and new, unknown or ad hoc word formations are not required (Gibbon et al., 1997). This procedure is convenient in languages with very small inflectional paradigms. An example of a language with few inflections is English, where, in general, three forms exist for nouns and four for verbs. English is therefore not a good example for illustrating inflectional morphology.

Agglutinative languages such as Turkish, Finnish or Hungarian, however, have complex inflectional and derivational morphology, with significantly more endings on all verbs, nouns, adjectives and pronouns. The number of endings increase the size of a basic vocabulary by a factor of thousands. Algorithmic morphological techniques have been developed for efficient composition of inflected forms and to avoid a finite but unmanageable explosion of lexicon size. Still, according to Althoff et al. (1996), these techniques have not been applied to any significant extent in speech technology.

[1] This framework is actually a second tier of the data flow. The user receives a black box providing linguistically sound and correctly recognized text. Inside the black box, the first tier performs the actual recognition, and the second tier carries out linguistic corrections and disambiguation.

In this paper, we describe the application of a new method based on morphology and partial parsing. This method uses a unified error model with flexible symbol mapping, facilitating the use of *any* linguistic module with traditional *orthographic* lexicons—for *any* recognition process (OCR, handwriting, speech recognition), even for highly inflectional languages. The integrated system uses our existing morpho-syntactic modules and lexicons.

1 The error model

The linguistic correction framework must be aware of three classes of error sources occurring in the input sequence: (1) poor or nonexistent segmentation, (2) underspecified symbols, (3) incorrectly recognised symbols.

The input sequence does *not* appear in the form of written text. It comprises of complex symbol codes in a normalized format, where the codes closely correspond to the signals recognized by the particular recognition process. In the case of OCR or handwriting recognition, this can be a shape code such as <lower> indicating a group of characters. With speech recognition, this is rather a phoneme code such as <e:>. (Here we use the notation of the proposed framework.) Standard orthographic characters may also appear in the input sequence.

With all types of recognition processes, there exists no one-to-one mapping between the symbols of the input sequence (the input alphabet) and the orthographic alphabet of the written text. The number of identified phonemes/phoneme complexes or characters/character complexes does not provide information about the number of characters to be used in the output text.[2] Unlike in two-level-morphology, the framework must provide *n*-to-*m* character (symbol) mapping, where $n \neq m$. Mapping between speech and text chunks of different length makes the system able to offer, for example, consonant sequences instead of affricates usually represented by single characters:

me{t^s,t^s:} ⇨ *metsz, metssz*

('engraves,slits,cuts' in affirmative and imperative)

[2] An example: in OCR outputs, the letter 'm' often occurs in place of the 'rn' sequence. The correction module must be able to transform single 'm'-s into 'rn'.

With continuous speech recognition (and, though less frequently, continuous handwriting recognition) it is even possible that a written segment boundary—such as the end of a word or a sentence—occurs within a symbol. The framework must be aware of these schemes as well.

The following sections present each error class with Hungarian examples to show the complexity of the linguistic model required by some languages.

2 The basics: symbol mapping

Atomic segments of input sequences are assumed to consist of (underspecified) symbols (phonemes/phoneme complexes, characters/character complexes). The correction framework must have a database of complex symbols—either phoneme codes or shape codes representing the classes of underspecified characters. An obvious approach to acquire a database of phonetic description of stems and suffixes (for morphological processing) is converting the existing (orthographical) lexicon. However, this conversion is very complicated and may result in an extremely large database. With speech recognition, for example, all orthographic representations must be converted into as many phonetic allomorphs as possible, on the basis of a grapheme-sequence-to-phoneme-sequence conversion. This set contains every allomorph where the first or the last phoneme of which is subject to contextual change. E.g. *két* ('two') is converted to *{ke:t, ke:t^y}*, because of palatalization before certain words, like *n^{y}u:l* ('rabbit').

ke:t^{y}n^{y}u:l ⇨ *két nyúl* ('two rabbits')

As the above method has some obvious disadvantages, we decided to separate the symbol mapping from the linguistic processes. We have created a database mapping the recognized symbols to all possible orthographical characters/character sequences. In this scheme, the framework creates several possible orthographical sequences from the input sequence (implemented internally as a directed acyclic graph for performance reasons). The correction framework then segments and validates each sequence using 'traditional' linguistic modules with the original orthographical lexicons. The conversion database uses a unified entry format suitable for all types of recognition processes. Example:

<ccs> ((<t>|((c|)c<s>)|)(c(<s>|)c<s>|ts))

This is a phoneme conversion entry. On the left side, a phonetic code is listed in the unified internal representation of the framework. (Note that this input symbol is the result of a mapping from the output of the recognition module.) On the right side, there is a directed acyclic graph (more or less a regular expression) describing *all* possible orthographic representations of the single phonetic entity.

This is the core idea of the framework: the separate conversion process provides for an open architecture where the framework can be attached to any recognition process, and even the linguistic modules are replaceable.

3 Morpho-lexical segmentation

For the simplest example, let us assume that the input sequence consists of phonetic symbols with no segmentation: however, pauses are indicated by the recognition process. The input sequence is processed symbol by symbol, and when the segmenter encounters a potential segment boundary, registers it and checks if the phonetic processor saw any pause, stress or other sign of segmentation at that particular position in the original speech signal. This might require some interaction with speech recognizer, but for the sake of simplicity, now we describe the operation of the linguistic subsystem only.

The original architecture design devises the framework as a feedback service, one requesting further information from the recognition source. In the current implementation, however, the correction framework can be separated from the recognition process, and provide corrected and disambiguated text without feedback to the recognition module.

In the analysis process of the unsegmented signal (see Figure 1), for example, the input slice *vonate:r* has three morpho-lexically equally likely segmentations: *von a tér, vonat ér, vonatér.* Either the acoustic signal contains information confirming or rejecting any of them; or all of them will be temporarily kept, and the segmentation process itself will filter them out later on. In Figure 1, after reading some further symbols from the input, it becomes clear that the only orthographically correct word boundary is between *vonat* and *érkezik* .

4 Underspecified forms

It is quite common that the recognition process cannot perfectly identify segments in the original signal source. These are the cases of *underspecification.* Let us assume that a speech recognition

Non-segmented phonetic input	**Proposed orthographic segmentation**
v	
vo	
von	*von*
vona	*von a*
vonat	*vonat*
vonate:	*vonaté*
vonate:r	*von a tér, vonat ér, vonatér*
vonate:rk	
vonate:rke	
vonate:rkez	
vonate:rkezi	*vonat érkezi*
vonate:rkezik	*vonat érkezik*
vonate:rkezika	*vonat érkezik a*
vonate:rkezikam	
vonate:rkezikama:	
vonate:rkezikama:š	*vonat érkezik a más*
vonate:rkezikama:šo	
vonate:rkezikama:šod	*vonat érkezik a másod*
vonate:rkezikama:šodi	*vonat érkezik a másodi*
vonate:rkezikama:šodik	*vonat érkezik a második*
vonate:rkezikama:šodikv	
vonate:rkezikama:šodikva:	
vonate:rkezikama:šodikva:g	*vonat érkezik a második vág*
vonate:rkezikama:šodikva:ga:	
vonate:rkezikama:šodikva:ga:n^{y}	*vonat érkezik a második vágány*
vonate:rkezikama:šodikva:ga:n^{y}r	
vonate:rkezikama:šodikva:ga:n^{y}ra	*vonat érkezik a második vágányra*

Figure 1. *Morpho-lexical segmentation of a Hungarian phonetic string*

process is unable to identify the value of the binary feature VOICED. In these cases, the linguistic subsystem attempts to find orthographically well-formed morpho-lexical constructions for both voiced and voiceless variants of the phoneme in question. In fact, underspecified forms of the input signal are represented either by lists of possible characters—like set representations in two-level morphology (Koskenniemi, 1983):

vona{t,d}e:rkezikama:šodi{g,k}{v,f}a:ga:nʸra

('train is arriving to the second platform')

or by underspecified feature complexes:

vonaDe:rkezikama:šodiGVa:ga:nʸra

where *D, G* and *V* are *d, g* and *v*, respectively, but not specified as voiced or voiceless.

5 Using higher-level linguistic processes

The linguistic correction framework operates rather inefficiently if it uses morpho-lexical processing only. This results in extreme ambiguity: numerous overgenerated orthographic patterns appear with grammatically incorrect segmentation. Thus the process must be improved by adding higher level linguistic analysis. Currently, the framework uses partial syntax similar to the mechanism applied in the Hungarian grammar checker module. This partial syntax describes particular syntactic phenomena in order to identify incorrect grammar beyond the word boundaries.

A more efficient post-processing filter is being developed by applying the *HumorESK* parser module (Prószéky, 1996). Figure 2 shows the possible

Input: *nʸelve:setitˢik:eti:rok*

1. **nyel vész e ti cikket ír ok*
2. **nyel vész e ti cikket írok*
3. **nyel vésze ti cikket ír ok*
4. **nyel vésze ti cikket írok*
5. **nyelv ész e ti cikket ír ok*
6. **nyelv ész e ti cikket írok*
7. **nyelvész e ti cikket ír ok*
8. **nyelvész e ti cikket ír ok*
9. **nyelvésze ti cikket ír ok*
10. **nyelvésze ti cikket írok*
11. *nyelvészeti cikket ír ok*
12. *nyelvészeti cikket írok*

('I am writing a linguistic paper.')

Figure 2. Syntactic filtering

segmentations of the morphology-only system. In this figure, an asterisk marks syntactically non-motivated word sequences filtered out by the partial syntax or the full parser—operating as a higher-level segmenter.

In the first 10 segmentations, the personal pronoun *ti* (2nd person, pl.) does not agree with either the verb *ír* (3rd person, sing.) or the verb *írok* (1st person, sing.). Syntactically the last two segmentations can be accepted (but semantically and according to topic-focus articulation, Nr. 11 is bizarre). In most cases it is true that the segmentation containing the longest matches in the input sequence is the best orthographical candidate.

6 Further development

Morpho-lexical and syntactic segmentation and correction can be very useful in improving the quality of 'traditional' recognition sources. However, it is important to emphasize that the proposed framework would only support existing recognition methods (e.g. likelihood-based mechanisms in speech recognition) rather than replacing them. The current breakdown of the framework makes no assumptions on the operation of the underlying recognition process, and does not prefer any methods to any other. In terms of architecture, the correction framework's operation is separated from the recognition module.

One of the aims of this project is, however, a better interaction between the linguistic and the recognition subsystem. As the first step, it requires a standard feedback interface (yet to be developed). Because the current implementation of the *MorphoLogic Recognition Assistant* framework does not make assumptions of the recognition subsystem, it cannot influence its operation. A standard feedback interface consists of a formalism for describing the interaction between a recognition source and the correction framework, regardless of the characteristics of the recognition subsystem. Stub modules must be developed to communicate with existing recognition systems.

An example for the dialogue between a phonetic and linguistic subsystem: first, a superficial acoustic-phonetic analysis offers some sequence of underspecified feature complexes, then the linguistic subsystem attempts to transform them into potential orthographically correct units with surface word boundaries. Finally, the phonetic system

controls whether which of the offered segmentation points can be confirmed acoustically.

7 Implementation

The first version of the MorphoLogic Recognition Assistant framework has been implemented along with a demonstration interface. This application takes symbolic codes of different recognized symbols (phonemes, OCR-read characters etc.), and provides orthographical output. It has been programmed in C++ using MS Visual Studio 6.0, and runs on 32-bit Windows systems. As service modules, the framework incorporates the *Humor* (morphological analyser), the *Helyesebb* (grammatical validator), and the *HumorESK* (full parser) technologies. With a standard programming interface, it is ready to be integrated with existing recognition systems.

Conclusion

This paper has introduced a framework for treating common error classes occurring in the output of various recognition sources. We have shown that different types of recognition sources share the same error types: namely, (1) poor or nonexistent segmentation, (2) underspecified and (3) incorrectly recognized symbols.

Our proposed solution is a post-processing phase performed on the output of the recognition source, where morpho-lexical and syntactic models validate (either accept or reject) different orthographical candidates derived from a single recognized symbol sequence.

The system is language independent and completely data-driven: by replacing the databases, the MorphoLogic Recognition Assistant is immediately ready to work with a different language. For the Humor system, descriptions exist for several languages (Hungarian, English, German, Spanish, Czech, Polish and Romanian). Syntax descriptions are under development for Hungarian and English (prototypes exist).

The proposed framework seems promising for continuous recognition systems. Its main advantage is the ease of application of any linguistic module, thanks to the separate symbol mapping process and the open architecture. However, we must emphasize again that the MorphoLogic Recognition Assistant supports *existing* recognition systems rather than replacing them.

Acknowledgements

This research was carries out within the framework of the IKTA-063/2000 Project supported by the Hungarian Ministry of Education.

References

Althoff, F., G. Drexel, H. Lüngen, M. Pampel & C. Schillo (1996). The treatment of compounds in a morphological component for speech recognition. In: D. Gibbon (ed.) *Natural language processing and speech technology*, 71-76, Mouton de Gruyter, Berlin, New York

Gibbon, D., R. Moore, R. Winski, eds. (1997). *Handbook of Standards and Resources for Spoken Language Systems.* Walter de Gruyter Publishers, Berlin & New York

Koskenniemi, K. (1983) *Two-level morphology: A general computational model for word-form recognition and production.* University of Helsinki, Department of General Linguistics, Helsinki, Finland

Nakayama, T. (1994). *Modeling Content Identification from Document Images.* Proceedings of the 4th Applied Natural Language Processing Conference, 22-27, Stuttgart, Germany

Prószéky, G. (1994). Industrial Applications of Unification Morphology *Proceedings of the 4th Conference on Applied Natural Language Processing*, 157–159, Stuttgart, Germany

Prószéky, G. (1996). Humor: a Morphological System for Corpus Analysis *First TELRI Seminar on Language Resources and Language Technology*, 149–158, Tihany, Hungary

Prószéky, G. (1996). Syntax As Meta-morphology *Proceedings of COLING-96*, Vol.2, 1123–1126, Copenhagen, Denmark

Prószéky, G. & B. Kis (1999). Agglutinative and Other (Highly) Inflectional Languages. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 261–268. College Park, Maryland, USA

Sibun, P. & Spitz, A. L. (1994). *Language Determination: Natural Language Processing from Scanned Document Images.* Proceedings of the 4th Applied Natural Language Processing Conference, 15-21,Stuttgart, Germany