Modeling of Polish language for Large Vocabulary Continuous Speech Recognition

Modelowanie języka polskiego dla ciągłego rozpoznawania mowy z uwzględnieniem obszernego zakresu słownictwa

Leszek Gajecki* and Ryszard Tadeusiewicz**

*Wyższa Szkoła Informatyki i Zarządzania, Rzeszów
**Akademia Górniczo-Hutnicza, Kraków
lgajecki@wsiz.rzeszow.pl
rtad@agh.edu.pl

ABSTRACT

The scope of this paper is modeling of Polish language for Large Vocabulary Continuous Speech Recognition (LVCSR). Language model is necessary part of LVCSR systems. For English language (for which probably the most solutions in field of speech recognition were created) researchers typically use a trigram model or other models which put accent on the order of words. However, in Slavic languages in many cases word order is not strict. This is the reason why models like trigrams etc. has difficulties in modeling of languages belonging to this group. Our idea is to employ Head-driven Phrased Structure Grammar (HPSG) for those purpose. We present here a very simple grammar based on HPSG idea, which can be the starting point for research on the use of HPSG in LVCSR for Slavic languages, particularly for Polish language.

STRESZCZENIE

Zakresem tej pracy jest modelowanie języka polskiego dla automatycznego rozpoznawania mowy (ARM) z użyciem dużego słownika. Model językowy jest niezbędnym elementem systemów ARM, w szczególności odnoszących się do dużego słownika. Dla języka angielskiego (dla którego opracowano prawdopodobnie najwięcej rozwiązań w dziedzinie rozpoznawania mowy) badacze najczęściej używają modelu trigramowego albo wykorzystują inne modele, które kładą nacisk na kolejność występowania słów w zadaniu. Jednakże w językach słowiańskich te modele są mniej przydatne, ponieważ tu w wielu przypadkach szyk zdania nie jest istotny. To sprawia, że modele takie jak model trigramowy mają trudności z reprezentacją cech języka słowiańskiego (na przykład polskiego) ważnych dla jego rozpoznawania. W pracy zaproponowano użycie gramatyki HPSG (*Head-driven Phrase Structure Grammar*). Przedstawiono bardzo prostą gramatykę bazującą na HPSG, która może być punktem wyjścia do badań nad użyciem HPSG w systemach ARM dla języka polskiego.

1. The aim of Language Model module

In a typical LVCSR system (for example [4, 5]) we can denote several blocks represented in figure 1.

SLTRA07ok:SLTRA07 2009-05-26 23:44 Strona 66

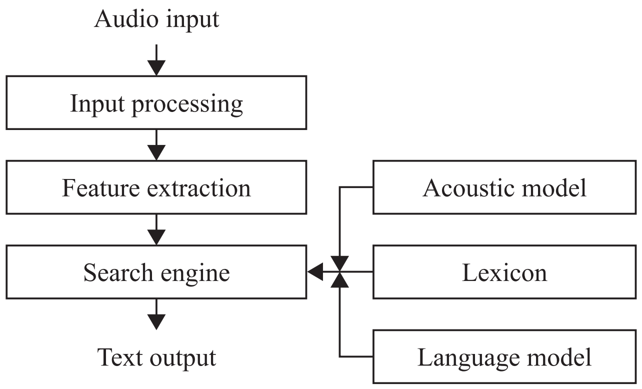

Figure 1. Architecture of a typical LVCSR system.

Firstly, we process the signal obtained from microphone (which involves e.g. amplification, normalization of spectrum). We do this in order to fulfil the requirements of the next module – A/D converter. Then digital signal is processed by Feature extraction module to get necessary features needed further. Search engine give us the most probable word sequence W_{opt} among possible word sequences $W_1^K = w_1, \dots , w_k$ given an acoustic feature vector $X = [x_1, x_2, \dots , x_n]$, so we can write:

$$W_{opt} = \arg\max_{W_1^K} P(W_1^K \mid X) .$$

To find this sequence such module uses 3 models: acoustic model, lexicon and language model. First of them – acoustic model is responsible for modelling the basic speech units (like phonemes, syllables, context-dependent phonemes etc.) Lexicon represents existing words – their pronunciation and spelling. The main field of our interest – language model gives an information (for example probability) that given words sequence represent a valid sentence. Here we can also take into account how common is given sentence.

2. Language model

2.1. Trigram statistical model

This model gives word probability as word probability for last $K - 1$ words multiplied by probability of new word w_k given all previous $N - 1$ words:

$$P(W_1^K) = P(W_1^{K-1})P(w_k \mid W_{K-N+1}^{K-1}) .$$

Here $N = 3$, so probability of new word w_k given all previous $N - 1$ words can be computed by counter function:

$$P(w_k \mid W_{K-2}^{K-1}) = \frac{c(w_{K-2}, w_{K-1}, w_K)}{c(w_{K-2}, w_{K-1})} .$$

2.2. Head-driven Phrase Structure Grammar (HPSG)

HPSG [1] is a linguistic theory that belongs to unification-based (or constraint-based) formalisms. It consists mainly of two parts: a small number of rules (which are general constraints) and large amount of lexical entries which explain word-specific dependencies. The example of a HPSG parsing is shown below.

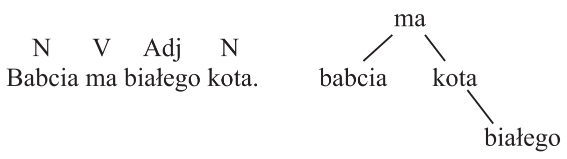

Figure 2. Example of a HPSG analysis.

Let's take the sentence *"Babcia ma białego kota"* (Grandmother has a white cat). Firstly, noun and adjective is connected using agreement schemata, which make phrase *"białego kota"*. This phrase has the same grammatical properties (gender, case,) as noun – head in this phrase. Then this phrase becomes an argument of verb-predicate *"ma"*. Such obtained phrase "ma kota białego" is predicate's group. Next level is connection of this group, with group of subject (only one word – noun *"babcia"*).

2.3. Simple grammar

We propose here a grammar for the Polish language which doesn't have big coverage, but can demonstrate the cooperation of HPSG with LVCRS systems. The constraints are:

1. Main element schemata
2. Word entry schemata.
3. Agreement schemata (only nouns and adjectives).

Explanation of above schemas (with respect to grammar of polish language) is described for example in [3].

The rules of lexical entries generation:

1. To represent the way that group of predicate connects with group of subject: verbs connect with subject as noun in nominative case, and arguments as nouns in other cases (one case per argument). In terms of HPSG this means that verb become predicate and with arguments constitute group of predicate. Noun, which satisfy requirements described above become subject and has the same properties as group of subject.
2. Nouns can connects with adjectives when they satisfy case agreement schemata), or they can have no arguments.
3. Other words has empty list of subject and arguments and modificators.

Our grammar has not got enough coverage, so mostly it cannot give us any information to prefer one sentence instead of another and we don't get any help in speech recognition. This is the reason why in contrast to HPSG parsing we don't answer if sentence is valid. Here we create the list of words from this sentence which each one was connected with at least one other word. In the same way we don't require, that all arguments of word has to be realized. This heuristic solution can give us partial information to find right sentence.

2.4. Application of HPSG

The idea of employing HPSG in Language Model module for improving results of LVCSR was presented for example by [2]. The N-best r list of recognized version of sentence is parsed by HPSG parser to check which sentence is valid linguistically. However our solution is slightly different. As we mentioned above, in case of our grammar we get response about partial validity, which we use to compute decreasing of sentence cost. Such result is list of all words, which each one connects with another word. After obtaining N_HPSG best hypotheses of sentence we parse them with our grammar. The number of obtained words and the length of their sequence for each hypothesis we take for computation of cost function:

$$h(W_1^K) = f(X, W_1^K) - HC \bullet \frac{l_p}{l} ,$$

where:

$f(X, W_1^K)$ – is cost function computed by search engine;
HC – is weight for HPSG module;

$l_p = \sum_{w_i \in WP_1^K} lenght(w_i)$ – is the total length of all words $w_i \in WP_1^K$ which connect with at least one word. The set $WP_1^K \subset W_1^K$ of such words is subset of the whole word sequence W_1^K;

lenght(w) – is the length of the word w;

$l = \sum_{w_j \in W_1^K} lenght(w_j)$ – is the total length of all words in word sequence W_1^K.

3. Experiments

3.1. Software

Here we describe the module of simplified software, which we constructed for checking influence of use the HPSG grammar in language model. Our software is divided into 2 parts:

1. Simulating modules of LVCSR system (Input processing, Feature extraction, Acoustic model).
2. Part which can be connected with real LVCSR system (Search engine, Lexicon, Language model).

The difference between typical and such simplified software is in the way how we simulate modules in part 1: We take the input sentence (from corpus). We concatenate its constituent words, so from this moment instead of speaking about voices we speak about letters. Next we add the "noise" – to simulate real recognition in meaning inaccurate recognition by acoustic model. This "noise" is changing of randomly chosen letters into next letter (we didn't go in simulation of similarities of acoustic properties of voices). The role of module of part 2 in such case is to separate words and in case of wrongly recognized voices (represented here by letters) we search for similar word.

As we see such approach cannot exactly represents behaviour of real LVCSR system. However it is enough to show work of language model.

3.2. Search Engine

This module searches for all possible words, which match to substring of letters in a given string. This substring starts from beginning of string. Its end we move one letter forward during every iteration. We also keep recognized words for each string and their total length to know form which position we have unrecognized substring. When we found word that match to that substring we add it to list of recognized words and update their total length. The function of cost f (X, W_1^K) – similar to cost function in typical LVCRS systems like [4] is given by equation

$$f\,(X,\, W_1^K) = CA\ (length(U)) + \log P(W_1^K) + CC \bullet K\ ,$$

where:

CC – cost for starting new word (it prevents from splitting longer words into smaller ones);
CA – weight for each unrecognized letter (when we finish creation list of words as recognized sentences);
W_1^K – considered sequence of *K* words;
length(*U*) – length of unrecognized string of letters at the end (string which don't match to any word);
P(W_1^K) – probability given by N-gram language model. There is possible to turn it off by substituting this value by 0).

We preferred sentences with smaller cost, so in each iteration we prune the worst solutions keeping BEAM_WIDTH best hypotheses. We set BEAM_WIDTH=1000 for our experiments, and BEAM_WIDTH=100 for tuning the parameters like CA and CC. Higher number make searching more detailed, but it consumes longer time of computations.

3.3. Lexicon

In our software lexicon module used for ASR has tree structure, which makes that finding right word has complexity *O*(*n log n*) (*n* is size of lexicon). It can give word requested by given letters, or next word in lexicographical order in case if given string doesn't represent word existing in lexicon.

3.4. Results

For experiments we used part of corpus of written polish language IPI PAN Corpus of Polish [6]. For "recognition" task we take subset containing 12 sentences – 160 words, closed vocabulary. Lexicon of HPSG was constructed from a subset of 2800 sentences (50k words), and trigram model was trained on 4500 sentences (58k words). Experiment with bigger trigram model (trained on 3,2 M words) doesn't bring changes.

At the first stage we turn weight parameters CA and CC without HPSG parsing. Next we performed the tests on the same set of 12 sentences, with "noise" in the same places. We searched for close to optimal value of HC weight.

We can notice, that all weights HC>0 make that the cost of word sequence that has words accepted by our parser is decreased (more decreased if the total length of such words is bigger percentage of the total length of all recognized words for this sequence). In case of HC<0 we prefer sequences that not match this criteria. Value HC=0 means that HPSG parsing is not taken into account.

Table 1. Word Error Rate(WER) for different weight HC in cost function

HC	WER [%] (+trigrams)	WER [%] (no trigrams)
0	77,5	77,5
1		77,5
10	79,3	77,5
100	85	77,5
1000	86,8	79,3
–2000	80	78,7

Higher values of HC can decrease the cost of worse word sequences, where longer words are split into shorter and therefore can also prefer them, which is the reason of increasing errors. We don't see improvement in using HPSG in comparison to ignoring information from HPSG parser. However in smaller values of HC results are the same.

The use of trigrams doesn't bring improvement too. We can notice, that combined application of HPSG and trigram model makes result worse (results in using of trigrams with HC>0 in comparison to experiments without trigrams in the same values of HC).

4. Conclusions

The approach presented here does not bring noticeable improvement, but there was no deterioration. We expect more developed HPSG grammar could prefer right sequences of words. We claim that developing of searching in lexicon similar words can improve quality of solutions. Here is limited ability to find similar word in case of significant deformation. Results can change by using real LVCSR system, because simplification in simulation of LVCSR presented here.

BIBLIOGRAPHY

[1] C.J. Pollard, I.A. Sag. *"Head-Driven Phrase Structure Grammar"*. The University of Chicago Press, Chicago1994.

[2] T. Kaufmann, B. Pfister. *"An HPSG Parser Supporting Discontinuous Licenser Rules"*. International Conference on HPSG, Stanford 2007.

[3] A. Przepiórkowski, A. Kupść, M. Marciniak, A. Mykowiecka. *"Formalny opis języka polskiego – Teoria i implementacja"*. Akademicka Oficyna Wydawnicza EXIT, Warszawa 2002.

[4] J. Duchateau. *"HMM based acoustic modelling in large vocabulary speech recognition"*. PhD thesis, Katholieke Universiteit Leuven 1998.

[5] *"Description of the ESAT speech recognition system"* – January 2006. (PSI – Speech Group Katholieke Universiteit Leuven, Belgium) http://www.esat.kuleuven.be/psi/spraak/.

[6] IPI PAN Corpus of Polish, http://korpus.pl.