LANGUAGE MODELING

Dr. Ivan Kraljevski, Dr. Frank Duckhorn

Fraunhofer Institute for Ceramic Technologies and Systems IKTS, Dresden

M.Sc. Isidor Konrad Meier

Chair of Communications Engineering, BTU Cottbis-Senftenberg, Cottbus

Daniel Sobe

Stiftung für das sorbische Volk, Bautzen

Table of contents

1 Introduction

Description: The current rule-based context-free grammars (CFG), which specify the possible inputs, limits speakers in the freedom of utterance formulations.
The concept of CFG grammas will be extended to language modeling with statistical language models and word classes in both cases. Such approach allows significantly more freedom in expression, where the speaker can speak the words from the vocabulary more flexibly without following the order as defined in a CFG grammar.

Terms like numbers, time, date, places, proper nouns will be specified with rule-based grammars for each word class. These and other word classes can be integrated into CFG grammars as well as in a statistical language model. Furthermore, it will be possible to combine models of different domains.

Tasks in this work-package:

1. Extension of the recognizer with word class grammars (time, date, numbers) as a prerequisite for speech applications with larger vocabularies.
2. Training a statistical language model with word classes (text data required)
3. Mixing domain-general and domain-specific models
4. Procedure for generating language models, which uses textual corpus with tagged word classes.

Deliverables: Word class grammars for predefined concepts (for application). Executable program to generate language models and configuration files, and short report with user guide.

Impact: Definition and generation of small vocabulary language models with configuration file. New language models can be used with the existing acoustic models (both, speaker independent and speaker dependent).

Prerequisites:

- sufficient text data (or specification in form of an ontology) with times, dates, and numbers.
- specification of the application

2 Language modeling

Depending on the intended speech application the language model can be defined either by context-free-grammar (CFG) or by a statistical language model (SLM). CFG grammars are appropriate for very limited vocabulary (few hundreds to thousand words) where the spoken utterance follow the expected order of words. In contrast, statistical language modeling (SLM) estimates the probability of word sequences based on N-gram statistics (unigrams, bigrams, trigrams, and more). Normally, in large vocabulary continuous speech recognition systems (LVCSR) a trigram back-off models are employed, and depending on the target domain, the vocabulary size could reach up to several hundred thousand words (dictation).
To train a SLM, a large amount of in-domain text data is required. Some examples of target domains are news, encyclopedic (Wikipedia), medicine, law, technology, etc. SLMs can recognize any sequence of words in any order if they are present in the vocabulary.
However, to achieve optimal performance in terms of word-error-rate (WER), the vocabulary should be restricted by the frequency of word occurrences. Usually, this is done by setting a criterion for a textual corpus, like top-N most frequent words (N=50000), or minimal count (i.e., a word occurred at least 5 times).
State-of-the-art performance with LVCSR is achieved in dictation systems where the acoustic and the language model are adapted to the specific speaker.

Language model adaptation is usually performed by updating (merging) the existing larger model with the weighted statistics of the N-grams calculated on a smaller amount of text.
The concepts of CFG and SLM with word-class modelling are be demonstrated in the following sections on the available textual data the Common-voice sentences and the sentences from the "Smartlamp" domain.

3 Context-free grammars

The existing CFG "Smartlamp" grammar is extended with new functionality (intents): making appointments (<APPOINTMENT>) and setting alarms (<WAKEMEUP>). Both new skills require handling of the following entities: numbers, date, and time.

Here is a specification example, that was accordingly implemented in the CFG grammar:

```
<WAKEMEUP> PROŠU (DATE) (|W|WE) (TIME) BUDŹIĆ
<WAKEMEUP> PROŠU BUDŹ MNJE (DATE) (|W|WE) (TIME)
<WAKEMEUP> LAMPA CHCU (DATE) (|W|WE) (TIME) STANYĆ

<APPOINTMENT> LAMPA (|ZAPISAJ (|MI)|ZARJADUJ (|MI)) NOWY TERMIN (DATE) (|W|WE) (TIME)
<APPOINTMENT> LAMPA (DATE) (|W|WE) (TIME) NJEZABYĆ NA (KOFEJ|NAKUP|DŹĚĆI|PÓŠTU|ZETKANJE|PARTY))
<APPOINTMENT> ČIN (DATE) TERMIN (|W|WE) (TIME)
<APPOINTMENT> ČIN TERMIN (DATE) (|W|WE) (TIME)
<APPOINTMENT> ČIN (DATE) (|W|WE) (TIME) TERMIN
<APPOINTMENT> DOPOMINAJ MNJE NA NARODNINY (DATE)
<APPOINTMENT> DOPOMINAJ MNJE NA SWJEDŹEŃ (DATE) (|W|WE) (TIME)
<APPOINTMENT> LAMPA DYRBJU (DATE) (|W|WE) (TIME) WOTJĚZDŹIĆ
```

We developed CFG grammar as finite-state-transducers (FST) that convert Upper Sorbian number, time, and date expressions into a proper numerical representation.
Each edge of a FST takes a substring of an expression as an input and converts it into an arithmetical operation consisting only of addition and multiplication operators.
The FSTs can be also nested, this incorporating smaller FSTs that represents more elementary expressions.

3.1 Word-class definitions

3.1.1 Numbers

We developed a basic numeral grammar NUM1-9 in the OpenFst-TextFile-Format[1], consisting of the cardinal numerals from 1 to 9. The basic grammar in incorporated to construct the grammar NUM1-99. Smaller grammars are combined to recursively construct larger grammars like NUM1-99, NUM1-999, NUM1-10^6, NUM1-10^{12} and NUM1-10^{15}.
Another important block in the construction of larger number grammars is NUM5-99. If the 2 digits before a decimal power's noun like "milion", "miliard", "bilion", ... are larger than 4, then the genitive plural "milionow"/"miliardow"/"bilionow"/... is used. Unlike, $3*10^6$ and $4*10^6$ call the nominative plural ("tři/štyri miliony"), $2*10^6$ the nominative dual ("dwaj milionaj") and $1*10^6$ the singular ("milion").
Another characteristic of Upper Sorbian numerals is that there are two words for the number 50. The expressions "pjećdźesat" (five tens) and "połsta" (half hundred) are arbitrarily interchangeable, even as subexpressions inside other numerals like of 51,150 or 50000.

[1] https://www.openfst.org/twiki/bin/view/FST/FstQuickTour#Creating%20FSTs%20Using%20Text%20Files%20f

We also developed some special numeral grammars like NUM0-23 and NUM0-59, which become incorporated in the time (clock) grammar.
Apart from cardinal numbers, we built ordinal numeral grammars ORD1-29, ORD30, ORD31, and a grammar for feminine ordinals ORD1-31f, which become incorporated in the date grammar.

3.1.2 Dates

For the date grammar, we again decided to represent the numerical meaning in a single number - the day count after New Year's Eve. Again, we also - and even mostly - use negative numbers. Since there are leap years, the day count after New Year's Eve is indefinite for any date from March to December. However, the day count till New Year's Eve is definite, so we use negative counts for March till December but positive counts for January and February. Hence, the output is always a number between -305 and 60.
Moreover, we developed two different date grammars for nominative and genitive case. Both cases are needed for some significant orders.
The date grammar is created out of an ordinal number representing the day and a word for the month. The ordinal number either comes from ORD1-29, ORD30, ORD31 or ORD1-31f, depending on the month.
For each month we essentially included 3 different names. A numerical name as the ordinal number of the month, a Gregorian name, and an older traditional name.

3.1.3 Time (Clock)

The time (clock) grammar converts time expressions into a numerical representation of time. As a representation we do not use the classical (hh:mm) format but rather just the count of minutes after midnight. We assume that this one-dimensional representation is not just easier to compute, but also easier to work with in the post-process for standard formatting.

To convert the time in a machine-readable standardized format (ISO 8601), the minute count m can easily be converted into hours and minutes with the following expression:

$$hh = \left\lfloor \frac{m}{60} \right\rfloor mod\ 24\ and\ mm = \ m\ mod\ 60$$

Note that the minute count output does not necessarily need to lie between 0 and 1440 but may also be negative. Still, the mod-operator in the above formulae will always led to an hour computation between 0-23.

We are considering two different types of clock time expressions:

1. One covers the accurate digital expressions, that are mostly used in official exact speech and for odd appointments of e.g., a bus or train departure. These expressions can be simply modeled out of NUM0-23 as the hour count, NUM0-59 as the minute count and "hodźin" as a connection word between the hour and the minute count.

2. The second type covers the more common everyday expressions like "tři štwórć na pjećich" (corresponding to "quarter to five"). We modeled this type of expressions as a construction out of 3 blocks.
    - The first block can represent the daytime (morning, noon, ...). It is important to include into the grammar, since it does influence the meaning, like in English "six in the morning" has a different meaning than "six in the evening".
    - The second block consists of any modifier of the time, like in English "X/quarter to past" or "half past" or even combinations out of those. We discussed with the client - the Załožba za serbski lud - to agree on which combinations of different modifiers should be covered. We agreed sub grammar presented on Figure 1.

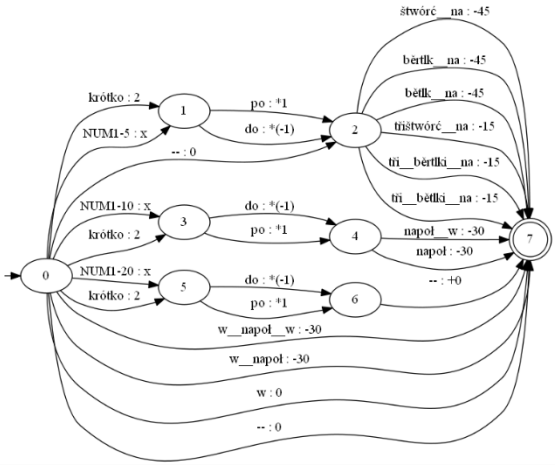

Figure 1. FST of the time modifiers

- The third part numerically represents the related hour. In Upper Sorbian everyday speech, a 12-hour format is used, so e.g., "dwěmaj" could either mean 2:00 or 14:00. The actual meaning must be determined based on the daytime.

3.2 Word-class CFG - "SmartLamp" demonstrator

3.2.1 Tools and configurations

The source metadata and the tools are in the folder:

```
$UASR_HOME-data/db-hsb-asr/grammatics/smartlamp_numbers/
```

The subfolder "grm" contains the grammar files that represents different building blocks for the numerals, time, and date, as well as the "SmartLamp" grammar that contains the newly added skills "appointment" and "wakemeup". The lexicon entries are generated with the BASGenerator python script as delivered.

Merging the dLabPro/UASR finite-state-grammar with the word-class grammars into a final-state-transducer format (OpenFST) is done by calling a python script:

```
cd recognizer
python ../grmmerge.py ../grm/smart_lamp_hsb_evl_fsg_num.txt
```

"pyhton grmmerge.py" without argument shows a short help about the possible options.

The script generates the language model (`smart_lamp_hsb_evl_fsg_num.txt_ofst.txt`) and the corresponding lexicon file (`smart_lamp_hsb_evl_fsg_num.txt_lex.txt`).

The next step is to create the recognizer configuration with:

```
dlabpro $UASR_HOME/scripts/dlabpro/tools/REC_PACKDATA.xtp rec ../default_fsg.cfg
```

The configuration file has following dLabPro/UASR configuration keys:

```
## UASR configuration file
## - Verbmobil database

uasr.db          = "db-hsb-asr";
uasr.exp         = "grammatics/smartlamp_numbers";

uasr.lm          ="fsg"
```

```
uasr.lx         ="$UASR_HOME-data/db-hsb-asr/grammatics/smartlamp_num-
bers/grm/smart_lamp_hsb_evl_fsg_num.txt_lex.txt";

uasr.lm.imp     ="$UASR_HOME-data/db-hsb-asr/grammatics/smartlamp_num-
bers/grm/smart_lamp_hsb_evl_fsg_num.txt_ofst.txt";

uasr.am.classes ="$UASR_HOME-data/db-hsb-asr/grammatics/smartlamp_numbers/model/classes.txt";
uasr.am.model="3_8";

uasr.out="$UASR_HOME-data/db-hsb-asr/grammatics/smartlamp_numbers/recognizer";

## EOF
```

The used acoustic model is a speaker-independent (3_8.hmm) as provided in the Work Package 1 ("Acoustic modeling"). It could be exchanged with a speaker adapted one for better performance.

The "`recognizer.cfg`" should have the matching acoustic model (3_8.gmm):

```
## UASR configuration file
## - Verbmobil database

## Recognizer configuration
data.feainfo = ./feainfo.object
data.gmm     = ./3_8.gmm
data.vadinfo =
data.sesinfo = ./sesinfo.object
search.typ   = tp
search.prn   = yes
search.tpprnw = 300
rej.typ = off
vad.nolimit = yes
```

3.2.2 Evaluation Results

A small set of fifteen audio examples of one speaker is used to evaluate and develop the grammar. The recognition of the files is called with:

```
recognizer -cfg recognizer.cfg ../sig/0002HSB_1_0.flst > results.log
```

And the recognition results redirected to a log file. The log wile can be used for more thorough analysis in terms of word error rates (WER) and character error rates (CER) for the semantics.

The output from the recognizer is postprocessed (with a proprietary software) using the Levenshtein algorithm to calculate the error rates.

Example of detailed results per utterance with the reference word and semantic sequences:

```
File: 0002HSB_1_000.wav
 Ref-Words: PROŠU POPOŁDNJU W TŘOCH BUDŹIĆ
 Res-Words: <PAU> <PAU> <PAU> PROŠU POPOŁDNJU TŘOCH BUDŹIĆ <PAU> <PAU> <PAU>
   Word-ER  20.0% C=4 I=0 D=1 S=0
 Ref-Sematic: PROŠU<TIME>+0+720+0+180</TIME>BUDŹIĆ
 Res-Sematic: PROŠU<TIME>+0+720+0+180</TIME>BUDŹIĆ
   Char-ER   0.0% C=36 I=0 D=0 S=0
```

```
File: 0002HSB_1_001.wav
 Ref-Words: ČIN NJEDŹELU TERMIN RANO DŹESAĆICH
 Res-Words: ČIN NJEDŹELU TERMIN RANO W DŹESAĆICH <PAU> <PAU> <PAU>
   Word-ER  20.0% C=5 I=1 D=0 S=0
 Ref-Sematic: ČIN<WDAY>+7</WDAY>TERMIN<TIME>+0+720+0-120</TIME>
 Res-Sematic: ČIN<WDAY>+7</WDAY>TERMIN<TIME>+0+720+0-120</TIME>
   Char-ER   0.0% C=49 I=0 D=0 S=0

File: 0002HSB_1_002.wav
 Ref-Words: LAMPA CHCU POPOŁDNJU ŠTYRJOCH STANYĆ
 Res-Words: <PAU> <PAU> LAMPA CHCU POPOŁDNJU ŠTYRJOCH STANYĆ <PAU> <PAU> <PAU>
   Word-ER   0.0% C=5 I=0 D=0 S=0
 Ref-Sematic: LAMPA_CHCU<TIME>+0+720+0+240</TIME>STANYĆ
 Res-Sematic: LAMPA_CHCU<TIME>+0+720+0+240</TIME>STANYĆ
   Char-ER   0.0% C=41 I=0 D=0 S=0

File: 0002HSB_1_003.wav
 Ref-Words: ČIN TERMIN SRJEDU WJEČOR W SEDMICH
 Res-Words: <PAU> <PAU> ČIN TERMIN SRJEDU WJEČOR SEDMICH <PAU> <PAU> <PAU>
   Word-ER  16.7% C=5 I=0 D=1 S=0
 Ref-Sematic: ČIN_TERMIN<WDAY>+3</WDAY><TIME>+720+0+420</TIME>
 Res-Sematic: ČIN_TERMIN<WDAY>+3</WDAY><TIME>+720+0+420</TIME>
   Char-ER   0.0% C=48 I=0 D=0 S=0

File: 0002HSB_1_004.wav
 Ref-Words: LAMPA PJATK DWĚMAJ NJEZABUDŹ NA KOFEJ
 Res-Words: <PAU> LAMPA PJATK DWĚMAJ NJEZABUDŹ NA KOFEJ <PAU> <PAU> <PAU> <PAU>
   Word-ER   0.0% C=6 I=0 D=0 S=0
 Ref-Sematic: LAMPA<WDAY>+5</WDAY><TIME>+0+0+120</TIME>NJEZABUDŹ_NA_KOFEJ
 Res-Sematic: LAMPA<WDAY>+5</WDAY><TIME>+0+0+120</TIME>NJEZABUDŹ_NA_KOFEJ
   Char-ER   0.0% C=59 I=0 D=0 S=0

File: 0002HSB_1_005.wav
 Ref-Words: LAMPA CHCU DRUHEHO MĚRCA RANO W DŹEWJEĆICH STANYĆ
 Res-Words: LAMPA CHCU DRUHO MĚRCA RANO W DŹEWJEĆICH STANYĆ <PAU> <PAU>
   Word-ER  12.5% C=7 I=0 D=0 S=1
 Ref-Sematic: LAMPA_CHCU<DATE>+2-306</DATE><TIME>+0+720+0-180</TIME>STANYĆ
 Res-Sematic: LAMPA_CHCU<DATE>+2-306</DATE><TIME>+0+720+0-180</TIME>STANYĆ
   Char-ER   0.0% C=60 I=0 D=0 S=0

File: 0002HSB_1_006.wav
 Ref-Words: LAMPA DYRBJU PJATK RANO W ŠTYRJOCH WOTJĚĆ
 Res-Words: <PAU> LAMPA DYRBJU PJATK RANO W ŠTYRJOCH WOTJĚĆ <PAU> <PAU> <PAU>
   Word-ER   0.0% C=7 I=0 D=0 S=0
 Ref-Sematic: LAMPA_DYRBJU<WDAY>+5</WDAY><TIME>+0+0+0+240</TIME>WOTJĚĆ
 Res-Sematic: LAMPA_DYRBJU<WDAY>+5</WDAY><TIME>+0+0+0+240</TIME>WOTJĚĆ
   Char-ER   0.0% C=56 I=0 D=0 S=0

File: 0002HSB_1_007.wav
 Ref-Words: PROŠU WUTORU RANO W SEDMICH BUDŹIĆ
 Res-Words: <PAU> PROŠU WUTORU RANO W SEDMICH BUDŹIĆ <PAU> <PAU> <PAU>
   Word-ER   0.0% C=6 I=0 D=0 S=0
 Ref-Sematic: PROŠU<WDAY>+2</WDAY><TIME>+0+0+0+420</TIME>BUDŹIĆ
 Res-Sematic: PROŠU<WDAY>+2</WDAY><TIME>+0+0+0+420</TIME>BUDŹIĆ
   Char-ER   0.0% C=49 I=0 D=0 S=0
```

```
File: 0002HSB_1_008.wav
 Ref-Words: LAMPA ZAPISAJ MI NOWY TERMIN DRUHI AWGUST WOSOMNAĆE HODŹIN PJAT NAĆE
 Res-Words: <PAU> LAMPA ZAPISAJ MI NOWY TERMIN DRUHI AWGUST WOSOMNAĆE HODŹIN PJAT NAĆE <PAU> <PAU> <PAU>
   Word-ER   0.0% C=11 I=0 D=0 S=0
 Ref-Sematic: LAMPA_ZAPISAJ_MI_NOWY_TERMIN<DATE>+2-153</DATE><TIME>+1080+0+15</TIME>
 Res-Sematic: LAMPA_ZAPISAJ_MI_NOWY_TERMIN<DATE>+2-153</DATE><TIME>+1080+0+15</TIME>
   Char-ER   0.0% C=70 I=0 D=0 S=0

File: 0002HSB_1_009.wav
 Ref-Words: DOPOMŃ MNJE NA NARODNINY PJATEHO NOWEMBRA
 Res-Words: <PAU> DOPOMŃ MNJE NA NARODNINY PJATEHO NOWEMBRA <PAU> <PAU> <PAU> <PAU>
   Word-ER   0.0% C=6 I=0 D=0 S=0
 Ref-Sematic: DOPOMŃ_MNJE_NA_NARODNINY<DATE>+5-61</DATE>
 Res-Sematic: DOPOMŃ_MNJE_NA_NARODNINY<DATE>+5-61</DATE>
   Char-ER   0.0% C=42 I=0 D=0 S=0

File: 0002HSB_1_010.wav
 Ref-Words: ČIN WOSMEHO JUNIJA PŘIPOŁDNJU W DWANAĆICH TERMIN
 Res-Words: <PAU> ČIN WOSMO JULIJA PŘIPOŁDNJU DWANAĆICH TERMIN <PAU> <PAU> <PAU> <PAU>
   Word-ER  42.9% C=4 I=0 D=1 S=2
 Ref-Sematic: ČIN<DATE>+8-214</DATE><TIME>+0+720+0+0</TIME>TERMIN
 Res-Sematic: ČIN<DATE>+8-184</DATE><TIME>+0+720+0+0</TIME>TERMIN
   Char-ER   3.9% C=50 I=1 D=1 S=0

File: 0002HSB_1_011.wav
 Ref-Words: PROŠU BUDŹ MNJE PÓNDŹELU RANO WOSMICH
 Res-Words: <PAU> PROŠU BUDŹ MNJE PÓNDŹELU RANO WOSMICH <PAU> <PAU> <PAU>
   Word-ER   0.0% C=6 I=0 D=0 S=0
 Ref-Sematic: PROŠU_BUDŹ_MNJE<WDAY>+1</WDAY><TIME>+0+0+0+480</TIME>
 Res-Sematic: PROŠU_BUDŹ_MNJE<WDAY>+1</WDAY><TIME>+0+0+0+480</TIME>
   Char-ER   0.0% C=53 I=0 D=0 S=0

File: 0002HSB_1_012.wav
 Ref-Words: NAKUP DŹĚĆI PÓŠTU ZETKANJE PARTY
 Res-Words: <PAU> <PAU> <PAU> NAKUP DŹĚĆI PÓŠTU ZETKANJE <PAU> KOFEJ <PAU> <PAU>
   Word-ER  20.0% C=4 I=0 D=0 S=1
 Ref-Sematic: NAKUP_DŹĚĆI_PÓŠTU_ZETKANJE_PARTY
 Res-Sematic: NAKUP_DŹĚĆI_PÓŠTU_ZETKANJE_KOFEJ
   Char-ER  15.6% C=27 I=0 D=0 S=5

File: 0002HSB_1_013.wav
 Ref-Words: LAMPA ZARJADUJ NOWY TERMIN PJEĆA DWACETO MĚRCA POPOŁDNJU W PJEĆICH
 Res-Words: <PAU> <PAU> LAMPA ZARJADUJ NOWY TERMIN PJEĆA DWACETO MĚRCA POPOŁDNJU W PJEĆICH <PAU> <PAU>
<PAU>
   Word-ER   0.0% C=10 I=0 D=0 S=0
 Ref-Sematic: LAMPA_ZARJADUJ_NOWY_TERMIN<DATE>+5+20-306</DATE><TIME>+0+720+0+300</TIME>
 Res-Sematic: LAMPA_ZARJADUJ_NOWY_TERMIN<DATE>+5+20-306</DATE><TIME>+0+720+0+300</TIME>
   Char-ER   0.0% C=73 I=0 D=0 S=0
File: 0002HSB_1_014.wav
 Ref-Words: DOPOMŃ MNJE NA SWJEDŹEŃ SOBOTU SEDMEHO SEPTEMBRA WJEČOR DŹESAĆICH
 Res-Words: DOPOMŃ MNJE NA SWJEDŹEŃ SOBOTU SEDMEHO SEDMO WJEČOR DŹESAĆICH <PAU> <PAU> <PAU> <PAU> <PAU>
   Word-ER  11.1% C=8 I=0 D=0 S=1
 Ref-Sematic: DOPOMŃ_MNJE_NA_SWJEDŹEŃ<WDAY>+6</WDAY><DATE>+7-122</DATE><TIME>+0+0-120</TIME>
 Res-Sematic: DOPOMŃ_MNJE_NA_SWJEDŹEŃ<WDAY>+6</WDAY><DATE>+7-184</DATE><TIME>+0+0-120</TIME>
   Char-ER   2.6% C=76 I=0 D=0 S=2
```

```
Total:
   Char-ER   1.1% C=789 I=1 D=1 S=7
   Word-ER   8.8% C=94 I=1 D=3 S=5
    CMD-ER  20.0% C=12 I=0 D=0 S=3
```

The recognition results show that word-class language modeling is successful. The misrecognitions are partially because to the different acoustic frontend and partially to the lexical modeling.

4 Statistical language models

In contrast of the CFG grammars which underperform when the spoken utterance is not defined in the grammar, the SLMs are capable to recognize words in arbitrary order even if they do not make any sense as long as if they are in the specified vocabulary.
Otherwise, they are considered as out-of-vocabulary (OOVs) words, and in recognition they are either substituted with the acoustical most similar or with the predefined OOV token (usually "<unk>").

In the following sections, we present guidelines how to create SLM with word-classes CFG grammars modelled as Finite-State-Transducers (FST).

4.1 Word-class tagging

Extensive description of word-class modeling is given in the section of the work package4 (WP4 section 2.3.3).

The corpus should have appropriate tags for the word-class grammars instead of the words that would be spoken to express, numbers, time, and dates.

For instance:

```
SWĚCA DAJ {PERCENT} PROCENTOW
```

Where the {PERCENT} is the named entity mapped with a grammar that define numbers from 1-100.

Textual corpus required for word-class statistical language modeling can be either collected and the words belonging to a predefined class (numerals, date, time) replaced with the appropriate tags enclosed with curly brackets ("{", and "}"). For instance, the string "28.02.2022" in the text would be replaced with the string "{DATE}".
Another way would be to use test generator defined with a CFG grammar and create sentences with already placed word-class tags.

The BASGenerator script can parse textual content with word-class tags and produce source files for the statistical language modelling with openFST. The word-class SLM is created in the same way as already described in the previous sections. The lexicons for all the word-classes should be included in the with the final lexicon.

4.2 Word-class SLM - "Smartlamp" demonstrator

An example is provided with the "word_class_lm" demo. Here the used top-level grammar is named as "NUM1-100" in OpenFst-TextFile-Format (Start_state End_state Input_token Output_token, and the last line is the last state). It could be arbitrarily re-named (e.g., "PERCENT_NUM") and the "<NUM> </NUM>" tags replaced by arbitrary alphanumeric string (e.g., "<PERCENT> </PERCENT>"):

NUM1-100.txt:

```
0 1 <eps>   <PERCENT>
1 2 NUM1-99 <eps>
1 2 STO     100
2 3 <eps>   </PERCENT>
3
```

4.2.1 Open FST Ngram

Common tool used to for FST based language modeling is the OpenFST and the nGramLibrary[1]. We provide step-by-step example how to create small bigram language models from the available textual corpora.
The normalized corpus and the vocabulary can be created by the BASGenerator (WP1 and 4) script with the appropriate input.

4.2.2 Training and evaluation of SLMs

Since the corpus is small with a small vocabulary, bigram SLM is created. Two files are required, the corpus (sl.corp) itself and the corresponding vocabulary (sl.vocab).

Step 1: Prepare vocabulary symbols

Create symbols from input vocabulary:

```
ngramsymbols --OOV_symbol="<unk>" sl.vocab sl.syms
```

Step 2: Prepare corpus

Compile the training corpus into FST archive (FAR file):

```
farcompilestrings --fst_type=compact --symbols=sl.syms --keep_symbols --unknown_symbol="<unk>"
sl.corp sl.far
```

Step 3: N-gram counting

Get the n-gram counts (2-grams):

```
ngramcount --order=2 sl.far sl.cnts
```

Step 4: FST model

Create a backoff LM FST model (sl.mod) with smoothing method:

[1] https://www.openfst.org/twiki/bin/view/GRM/NGramLibrary

```
ngrammake --backoff --method=witten_bell sl.cnts sl.mod
```

Optional: Pruning

Pruning is necessary to reduce the model size (in case of a large vocabulary, >50K words) by removing the less probable N-grams.

Prune the model with different methods (relative_entropy):

```
ngramshrink --method=relative_entropy --theta=1.0e-7 sl.mod sl.pru
```

Optional: Export in ARPA

Create ARPA format that can be used in other frameworks:

```
ngramprint --ARPA sl.pru sl_pru.ARPA
ngramprint --ARPA sl.mod sl.ARPA
```

Optional: Import from ARPA

Import form ARPA format LM created with different tools (e.g., IRSLM, MITLM, SRILM, KenLM, etc.)

```
ngramread --ARPA sl.arpa sl.mod
```

Optional: Apply FST LM

Apply model to input text.

```
ngramapply sl.mod sl.far | farprintstrings --print_weight
```

Step 6: Model evaluation

Evaluate model on the SL corpus in FAR format. Lower perplexity[1] is better.

```
ngramperplexity --OOV_symbol="<unk>" sl.mod sl.far
ngramperplexity --OOV_symbol="<unk>" sl.pru sl.far
```

4.2.3 Merging and evaluation of two SLMs

Merge two different language models, starting from text corpus.

```
ngrammerge --help
```

The corpuses should be in the format where sentences with white spaces as word delimiters are ending with a newline. In the example we merge the SL (SmartLamp) as specialized domain with the adaptation content (sentences from the CV and HSB corpus) as general domain.

[1] https://en.wikipedia.org/wiki/Perplexity

Step 1: Make symbol tables:

```
ngramsymbols ad.corp ad.syms
ngramsymbols sl.corp sl.syms
```

Step 2: Convert to FAR archives:

```
farcompilestrings --fst_type=compact --symbols=ad.syms --keep_symbols --unknown_symbol="<unk>"
ad.corp ad.far
farcompilestrings --fst_type=compact --symbols=sl.syms --keep_symbols --unknown_symbol="<unk>"
sl.corp sl.far
```

Step 3: Make the N-gram counts:

```
ngramcount --order=2 ad.far ad.cnts
ngramcount --order=2 sl.far sl.cnts
```

Step 4: Merging

Merge the counts with default parameters, compile and prune the model:

```
ngrammerge ad.cnts sl.cnts merged.cnts
ngrammake merged.cnts merged.mod
ngrammake sl.cnts sl.mod
ngrammake ad.cnts ad.mod
```

But, also possible to merge the FSTs directly (alpha and beta weights):

```
ngrammerge --alpha=3 --beta=2 ad.mod sl.mod > merged.mod
```

Optional: Pruning

Pruning is too extreme in the case of such small models:

```
ngramshrink --method=relative_entropy --theta=.00015 merged.mod merged.pru
```

Step 5: Evaluation

Evaluate the models:

- With matching domains and no OOVs:

```
ngramperplexity ad.mod ad.far
ngramperplexity sl.mod sl.far
```

- Nonmatching domains, without OOVs:

```
ngramperplexity ad.mod sl.far
ngramperplexity sl.mod ad.far
```

- Merged model:

```
ngramperplexity --OOV_symbol="<unk>" merged.mod ad.far
ngramperplexity --OOV_symbol="<unk>" merged.mod sl.far
```

4.3 Tools and configurations

The source metadata and the tools are in the folder:

```
$UASR_HOME-data/db-hsb-asr/grammatics/word_class_lm/
```

The already created language model in FST format (wcls.fst) must be converted to OpenFst-TextFile-Format format (example with "{PERCENT}"):

```
cd lm
fstprint.exe wcsl.fst | \
sed -e 's/<epsilon>/<eps>/g' | \
sed -e 's/{PERCENT}\(.*\){PERCENT}/NUM1-100\1<eps>/' > wcsl.txt
```

Hier fehlt "grmmerge.py -dbg -lmw 10 -ofstin wcsl.txt".
NUM1-100.txt muss im gleichen Verzeichnis wie wcsl.txt sein!

Merging the language model with the word-class grammars into a final-state-transducer format (OpenFST) is done by calling a python script:

```
python ../grmmerge.py -lmw 10 wcsl.txt
```

Important: NUM1-100.txt must be in the same folder as wcsl.txt!

The next step is to create the recognizer configuration with:

```
cd ../recognizer
dlabpro $UASR_HOME/scripts/dlabpro/tools/REC_PACKDATA.xtp rec ../default_lm.cfg
```

The configuration file has following dLabPro/UASR configuration keys:

```
uasr.db         = "db-hsb-asr";
uasr.exp        = "grammatics/word_class_lm";
uasr.lm         = "fsg"
uasr.lx         = "$UASR_HOME-data/db-hsb-asr/grammatics/word_class_lm/lm/wcls.ulex";
uasr.lm.imp     = "$UASR_HOME-data/db-hsb-asr/grammatics/word_class_lm/lm/wcsl.txt_ofst.txt";
uasr.am.classes = "$UASR_HOME-data/db-hsb-asr/grammatics/word_class_lm/model/classes.txt";
uasr.am.model   = "3_8";
uasr.out        = "$UASR_HOME-data/db-hsb-asr/grammatics/word_class_lm/recognizer";
```

4.4 Evaluation results

A small set of 6 audio examples of one speaker (0006HSB_3) is used to evaluate and develop the grammar. The recognition of the files is called with:

```
recognizer -cfg recognizer.cfg ../sig/0006HSB.flst
```

And the recognition results redirected to a log file. The log wile can be used for more thorough analysis in terms of word error rates (WER) and character error rates (CER) for the semantics. The output from the recognizer is postprocessed (with a proprietary software) using the Levenshtein algorithm to calculate the error rates. Example of detailed results per utterance with the reference word and semantic sequences:

```
File: 0006HSB_3_251_2.wav
 Ref-Words: SWĚCA DAJ PJAT NAĆE PROCENTOW
 Res-Words: SWĚCA DAJ PJAT NAĆE PROCENTOW
   Word-ER   0.0% C=5 I=0 D=0 S=0
 Ref-Sematic: SWĚCADAJ<NUM>+15</NUM>PROCENTOW
 Res-Sematic: SWĚCADAJ<PERCENT>+15</PERCENT>PROCENTOW
   Char-ER  43.8% C=28 I=10 D=2 S=2
```

```
File: 0006HSB_3_253_2.wav
 Ref-Words: LAMPA PJEĆ DŹESAĆ PROCENTOW
 Res-Words: LAMPA PJEĆ DŹESAĆ PROCENTOW
   Word-ER   0.0% C=4 I=0 D=0 S=0
 Ref-Sematic: LAMPA<NUM>+50</NUM>PROCENTOW
 Res-Sematic: LAMPA<PERCENT>+5</PERCENT><PERCENT>+10</PERCENT>PROCENTOW
   Char-ER 125.0% C=24 I=31 D=2 S=2

File: 0006HSB_3_255_2.wav
 Ref-Words: LAMPA PJEĆ A TŘI CEĆI PROCENTOW PROŠU
 Res-Words: LAMPA PJEĆ A TŘI CEĆI PROCENTOW PROŠU
   Word-ER   0.0% C=7 I=0 D=0 S=0
 Ref-Sematic: LAMPA<NUM>+5+30</NUM>PROCENTOWPROŠU
 Res-Sematic: LAMPA<PERCENT>+5+30</PERCENT>PROCENTOWPROŠU
   Char-ER  38.9% C=32 I=10 D=2 S=2

File: 0006HSB_3_260_2.wav
 Ref-Words: SWĚCA ŠTYR CEĆI PROCENTOW
 Res-Words: SWĚCA ŠTYR CEĆI PROCENTOW
   Word-ER   0.0% C=4 I=0 D=0 S=0
 Ref-Sematic: SWĚCA<NUM>+0+40</NUM>PROCENTOW
 Res-Sematic: SWĚCA<PERCENT>+0+40</PERCENT>PROCENTOW
   Char-ER  45.2% C=27 I=10 D=2 S=2

File: 0006HSB_3_263_2.wav
 Ref-Words: SWĚCA DWA CEĆI PROCENTOW PROŠU
 Res-Words: SWĚCA DWA CEĆI PROCENTOW PROŠU
   Word-ER   0.0% C=5 I=0 D=0 S=0
 Ref-Sematic: SWĚCA<NUM>+0+20</NUM>PROCENTOWPROŠU
 Res-Sematic: SWĚCA<PERCENT>+0+20</PERCENT>PROCENTOWPROŠU
   Char-ER  37.8% C=33 I=10 D=2 S=2

Total:
   Char-ER  55.5% C=144 I=71 D=10 S=10
   Word-ER   0.0% C=25 I=0 D=0 S=0
    CMD-ER 100.0% C=0 I=0 D=0 S=5
```

5 Deliveries

Two examples that include word-classes grammars, expanding the functionality and flexibility of the "SmartLamp" demonstrator. They should be placed in custom folder and the UASR config files accordingly updated.

5.1 CFG with date, time, and number grammars

The folder contains the following data:

```
default_fsg.cfg         - configuration file
grm                     - grammar folder
grm2ofst.xtp            - dlabpro grammar to FST conversion script
grmmerge.py             - script to merge grammars
model                   - default acoustic model
recognizer              - recognizer with configurations
sig                     - audio and reference text (*_is.txt) and semantics (*_os.txt)
trl                     - transliterations
```

5.2 SLM with number grammars

The folder contains the following data:

```
default_lm.cfg          - configuration file
grmmerge.py             - script to merge grammars and statistical language model
lm                      - language model and grammars
model                   - default acoustic model
recognizer              - recognizer with configurations
sig                     - audio and reference text (*_is.txt) and semantics (*_os.txt)
sources                 - source files to build a language model from textual corpus
trl                     - transliterations
```