Integrating natural language processing to language documentation

Course: NLP for Endangered Languages of the Amazon. From a Uralic perspective. Lecture 4.

Jack Rueter, Niko Partanen,
Mika Hämäläinen & Khalid Alnajjar

File formats

- Most of the software used in language documentation uses XML
- This has been the trend for the last 15 years
- This works well, but there are a few issues to consider:
  - XML can be difficult to edit and process, and linguists are rarely taught to do that
  - Some XML structures are particularly cumbersome
  - The fact that something is in XML doesn’t guarantee that all information is correctly connected
- Examples:
  - In ELAN it is possible to create hierarchies that look alright, but the internal structure is messy
  - ELAN XML stores the tier relations in a way that can be challenging to parse
  - FLEx stores the main work file in a large zipped XML file, which is very complex
  - **.fwdata and .fwdatabackup can be renamed to .zip and opened**

- Example from a very easily understandable XML is what Lameta produces
- Also LIFT export from FLEx is pretty good
- XML is a good format, but we usually do not want to see it or touch it
- Ideally we can use it and move the information we want to other environments

- Next we go through just some basic XML concepts

Basic XML structure

<entry> ← Opening node
</entry> ← Closing node

Basic XML structure

<entry> ← Opening node
  <lemma></lemma> ← Child node
</entry> ← Closing node

Basic XML structure

<entry number=”1”> ← Opening node with a tag that has a value “1”
  <lemma></lemma> ← Child node
</entry> ← Closing node

Basic XML structure

<entry number=”1”> ← Opening node with a tag that has a value “1”
<lemma>kiritãtxi</lemma> ← Child node with **text**
</entry> ← Closing node

Basic XML structure

<entry number="1"> ← Opening node with a tag that has a value 1
<lemma pos="N">kiritãtxi</lemma> ← Child node with **text** and tag for POS
</entry> ← Closing node

Basic XML structure

<entry number="1"> ← Opening node with a tag that has a value “1”
<lemma pos="N">kiritãtxi</lemma> ← Child node with **text** and tag for POS
<translation language="eng">nose</translation> ← Child node with translation
</entry> ← Closing node

Basic XML structure

<entry number="1"> ← Opening node with a tag that has a value “1”
<lemma pos="N">kiritãtxi</lemma> ← Child node with **text** and tag for POS
<translation language="eng">nose</translation> ← Child node with translation
<translation language="fin">nenä</translation> ← Child node with translation
</entry> ← Closing node

Basic XML structure

```
<entry number="1">
<lemma pos="N">kiritãtxi</lemma>
<translation language="eng">nose</translation>
<translation language="fin">nenä</translation>
</entry>

<entry number="2">
<lemma pos="N">namatxi</lemma>
<translation language="eng">mouth</translation>
<translation language="fin">suu</translation>
</entry>
```

Basic XML structure

```
<?xml version="1.0"?>                          ← XML declaration
<dictionary language="apu">                    ← Highest level node
<entry number="1">
<lemma pos="N">kiritãtxi</lemma>
<translation language="eng">nose</translation>
<translation language="fin">nenä</translation>
</entry>

<entry number="2">
<lemma pos="N">namatxi</lemma>
<translation language="eng">mouth</translation>
<translation language="fin">suu</translation>
</entry>
</dictionary>
```

File formats

- Main alternative file format we could use is JSON
- In many instances even a spreadsheet is already a good alternative
  - For example, with basic session level metadata this can work just fine
  - We should use as complicated structures *as we need*

- Example we just discussed would be in JSON:
- Even without more specific new technology we can do a lot

```
[{'lemma': 'kiritãtxi',
  'pos': 'N',
  'translation_eng': 'nose',
  'translation_fin': 'nenä'},
 {'lemma': 'namatxi',
  'pos': 'N',
  'translation_eng': 'mouth',
  'translation_fin': 'suu'}]
```

Dictionary example

FLEx allows exporting LIFT XML

It is little bit more complicated than our previous example

- Lemma / form
- Type
- Pronunciation
- POS
- Translations
- Example
- Example translations

```
<entry dateCreated="2014-12-29T22:04:11Z"
dateModified="2015-02-15T00:53:44Z" id="1" guid="2">
<lexical-unit>
<form lang="ktn"><text>dapy</text></form>
</lexical-unit>
<trait  name="morph-type" value="stem"/>
<pronunciation>
<form lang="ktn"><text>da.pɨ </text></form>
</pronunciation>
<sense id="7e14b6f7-fb98-4e4a-95d5-9cf54741443d">
<grammatical-info value="Noun">
</grammatical-info>
<gloss lang="en"><text>bunch (of fruit)</text></gloss>
<gloss lang="pt"><text>penca (de frutas)</text></gloss>
<example>
<form lang="ktn"><text>yhira yno myhint dapydnat asyrytyty</text></form>
<translation type="Free translation">
<form lang="pt"><text>me dá uma penca de bananas</text></form>
</translation>
</example>
</sense>
</entry>
```

Reading LIFT to a very preliminary JSON

A small example script in Python

We parse each entry, and if there is a translation we keep it. We could take both translations, but that is easy to change.

{'form': 'ambi',
'pronunciation': 'ã.mbi',
'translation_en': 'house',
'example': 'taso nakaw̆at ambi',
'example_translation': 'the man made the house'}

Writing JSON into Markdown

Markdown is a simple **syntax** for writing ****documents****.

It can be easily converted to Word document, HTML or PDF. If we need just simple formatting, it's a very good and general choice.

```
<lexical-unit>
<form lang="ktn"><text>ambi</text><
</lexical-unit>
<trait  name="morph-type" value="st
<note type="bibliography">
<form lang="pt"><text>LANDIN, D. Di
Disponível em: &lt;http://www01.sil
</note>
<note>
<form lang="pt"><text>ami (ortograf
</note>
<pronunciation>
<form lang="ktn"><text>abmbi</text>
</pronunciation>
```

```
{'form': 'ambi',
'pronunciation': 'ã.mbi',
'translation_en': 'house',
'example': 'taso nakaŭat ambi',
'example_translation': 'the man
made the house'}
```

```
**ambi** \[ã.mbi\] house : *taso
nakaŭat ambi* the man made the
house
```

ambi [ã.mbi] house : *taso nakaŭat ambi* the man made the house

ambi [abmbi] house

ambidna [ã.mbi.dnã] to live : *tasoojdn tykiri naambidnat taso* when th
has his home

ambigng [abmbigŋ̃] to build a house

ambik [abmbik] to sit

ambipidna [abmbipidnã] lazy

ambipitydnom [abmbipitɨdnõm] wall

ambipy [abmbipɨ] ceiling, roof

ambo [abmbo] to lay down

ambopa [abmbopa] stairs

Currently ignored features

- Related entries should be presented together
- Features that have missing values could be saved separately elsewhere
- One could display both English and Portuguese translations
  - Or have two separate exports
- If the word has no pronunciation, there could be rules to generate it, with some highlighting that this is not verified

Appendix 1: Lexicon & Examples

Niko Partanen & Alexandra Kellner

аддзывны to see (once); to glance at **TV**

1. воас нин асыв, колӧ сетныс рисунокйӧстӧ царлы; иваныд друг тай босьтас, **аддзылас** аврамыдлісь кыдзи сія рисуйтӧма. EVV (1998)

аддзыны to see **TV**

1. Сія этатшӧ рузь кӧ йӧрысь **адьдзас**, вермас ні ветлыны. Сія OKZR, page 247
2. las i, ovli̮vlę, **ad́d́źas** lebę́ t́śę̆msę), sija kuńi₍t́śa·i̮s ńe-ki̮t́t́śę òz ST III, page 358
3. kor munę mami̮s, kupet́ś vi̮dli̮·las ţşet́veri·ksę i **ad́d́źas** d́éńgaję·zdę. FF (1916), page 151
4. mužik menam **ad́d́źas** i ki̮las. i voi̮i·sni̮s baba da drug. pondi·s- FF (1916), page 166
5. voni̮. vę₍d́źę i med bęrja·i̮s śuvśas. i sija **ad́d́źas**, štę ti̮tęn i₍d̦ži̮d FF (1916), page 171
6. **ad́d́źas** i vętas. ośki̮s ponji̮s vi̮lę kujimjez lebę́ t́śi̮·las i mitréji̮s FF (1916), page 179
7. aja-pii̮a męda·sni̮ ari̮n oš vini̮. i sije·zda **ad́d́źa·sni̮** oš-gu. FF (1916), page 182
8. i jeśĺi **ad́d́źas**, to pondas gorzi̮·ni̮: «i̮ręmĺi, i̮ręmĺi!» śetas znak, FF (1916), page 200
9. **ad́d́źa·sni̮**, sija vę₍d́źę oz d́źebśi̮ i bęra męda·sni̮ korśi̮·ni̮, i ad́- FF (1916), page 200
10. ad́d́źa·sni̮, sija vę₍d́źę oz d́źebśi̮ i bęra męda·sni̮ korśi̮·ni̮, i **ad́-**
d́źa·sni̮-si, tężę ponda·sni̮ gorzi̮·ni̮. jeśĺi śurę-kę, kodi kutaśę·ni̮, seś FF (1916), page 200
11. mami̮s i̮stas šęr-kos ni̮ lęs i sija bara **ad́d́źas** zarńi tupi̮ĺęs. FF (1916), page 206
12. i med pońi ni̮lęs i̮stas. sija bara **ad́d́źas** zarńi tupiĺęs. vęt- FF (1916), page 207
13. vi̮liś **ad́d́źas** jašţşik. vęśtas jašţşik i ad́d́źas med i₍d̦ži̮d ni̮lęs. FF (1916), page 207
14. vi̮liś ad́d́źas jašţşik. vęśtas jašţşik i **ad́d́źas** med i₍d̦ži̮d ni̮lęs. FF (1916), page 207

ELAN integration of NLP tools

- ELAN file is an XML file and can be edited directly
- It is possible to run scripts to the ELAN files, or use analysers as plugins
- In our projects we have used analysers with Komi and Saami languages
- Current situation:
  - Tokenization
  - Lemmatization
  - Morphological analysis
  - Syntactic analysis (for Northern Saami)
- Everything is fully available, but works well for Uralic languages
  - https://github.com/langdoc/elan-fst

Example

Our script is currently running at server in Helsinki:

http://193.167.189.183/elan-fst/

Works with two ELAN templates:

- One used in our Komi project
- One used in Oulu University's Saami work

Besides annotated file, we also get list of unknown transcribed words.

- This builds a very efficient workflow!

ELAN annotation pipeline

File processed successfully

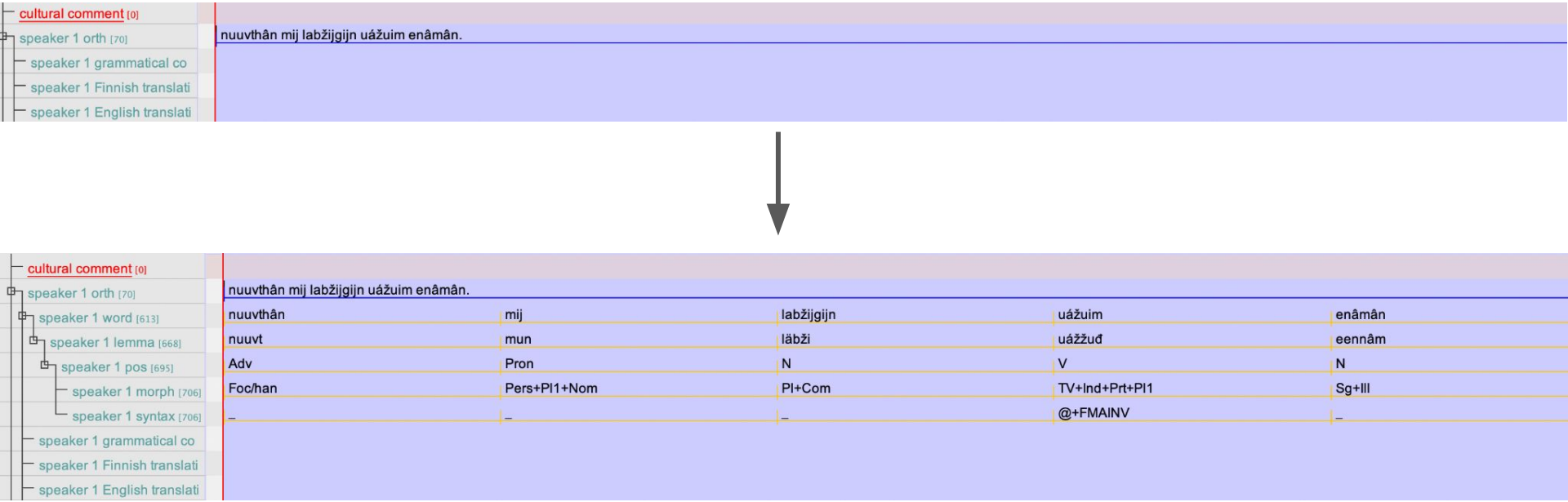
File Name: 13755_2az-soft-haired-sister-song.eaf

Download annotated file

Unrecognized forms

Form	Count
сэтче	5
кучисныс	5
ырген	4
ооны	4
чомъе	3
нырсьыс-вомсьыс	3
какен	3
тшесьтыс	2
тае	2
сюдбей	2
суутэдіс	2
сима	2
сеччы	2
саюйяс	2
лямкооны	2
кӧрныссэ	2
кыызы	2
ку-	2
коймед	2
казёолан	2
зыредіс	2
дадьбӧр	2

Further example: Writing annotations directly to ELAN

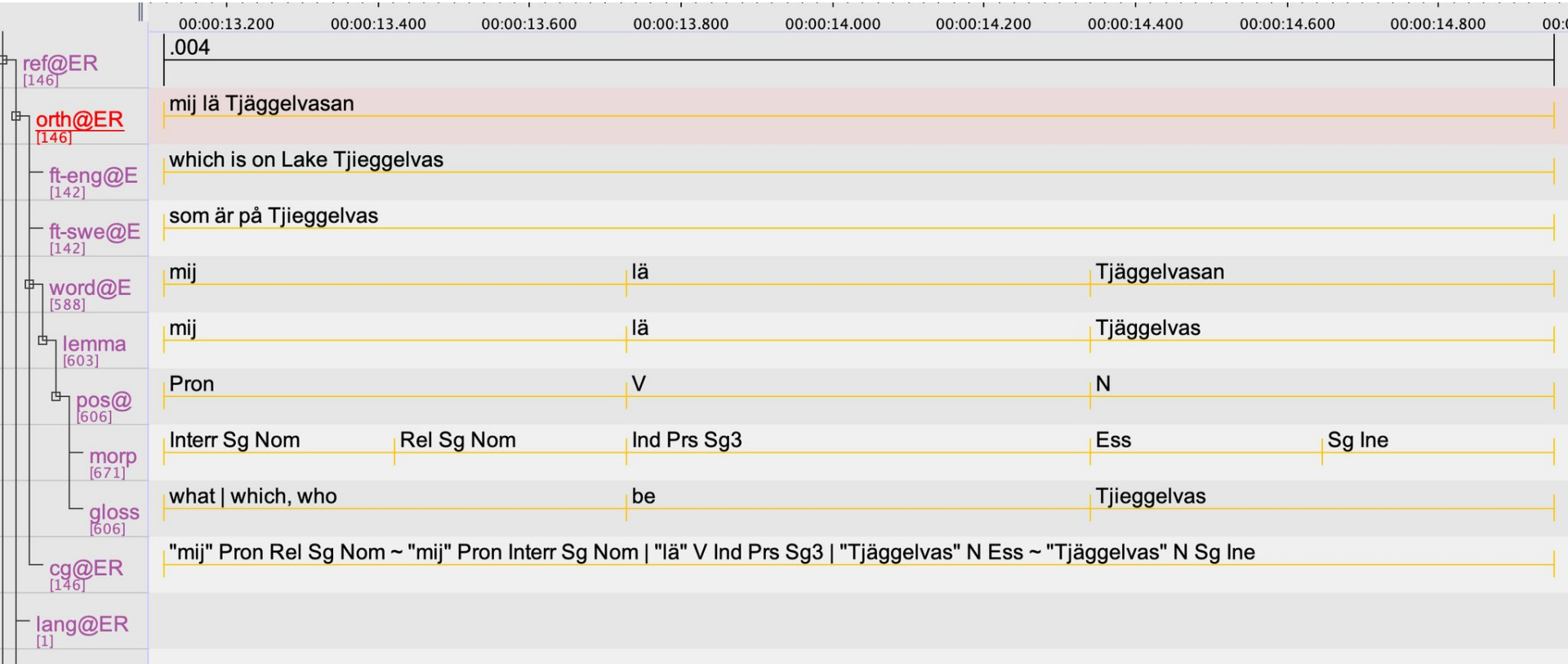


“So in this very way we got them ashore with belts”

Source: Giellagas Corpus of Spoken Saami Languages (Inari Saami portion)

Our repository for related work: https://github.com/langdoc/elan-fst

Pite Saami example from Joshua Wilbur



Discussion

This approach is not directly compatible with the work in FLEx

Meaning mainly that these approaches have not crossed a lot

A lexicon stored in FLEx could, however, be integrated into computational infra

The solution would essentially work like in conversion example above.

- LIFT XML > JSON / other XML > Ve’rdd

This way one could maintain the dictionary in FLEx, but benefit from NLP solutions

ASR (automatic speech recognition) – situation today

- There are continuously new speech recognition frameworks
- They change every year, and the progress is fast
- Currently best results when there is only one speaker
- We usually measure the quality with label, phoneme or character error rate

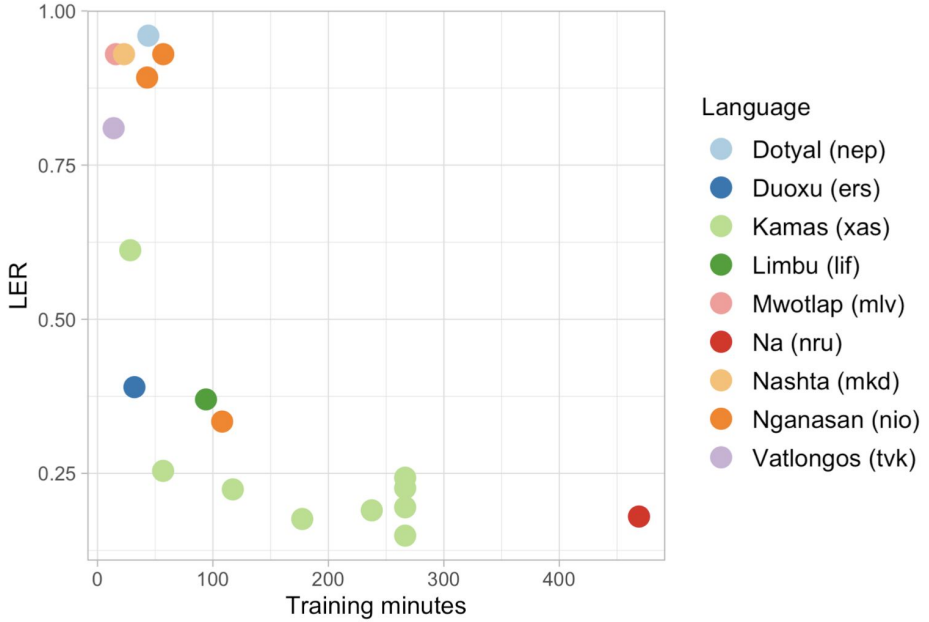


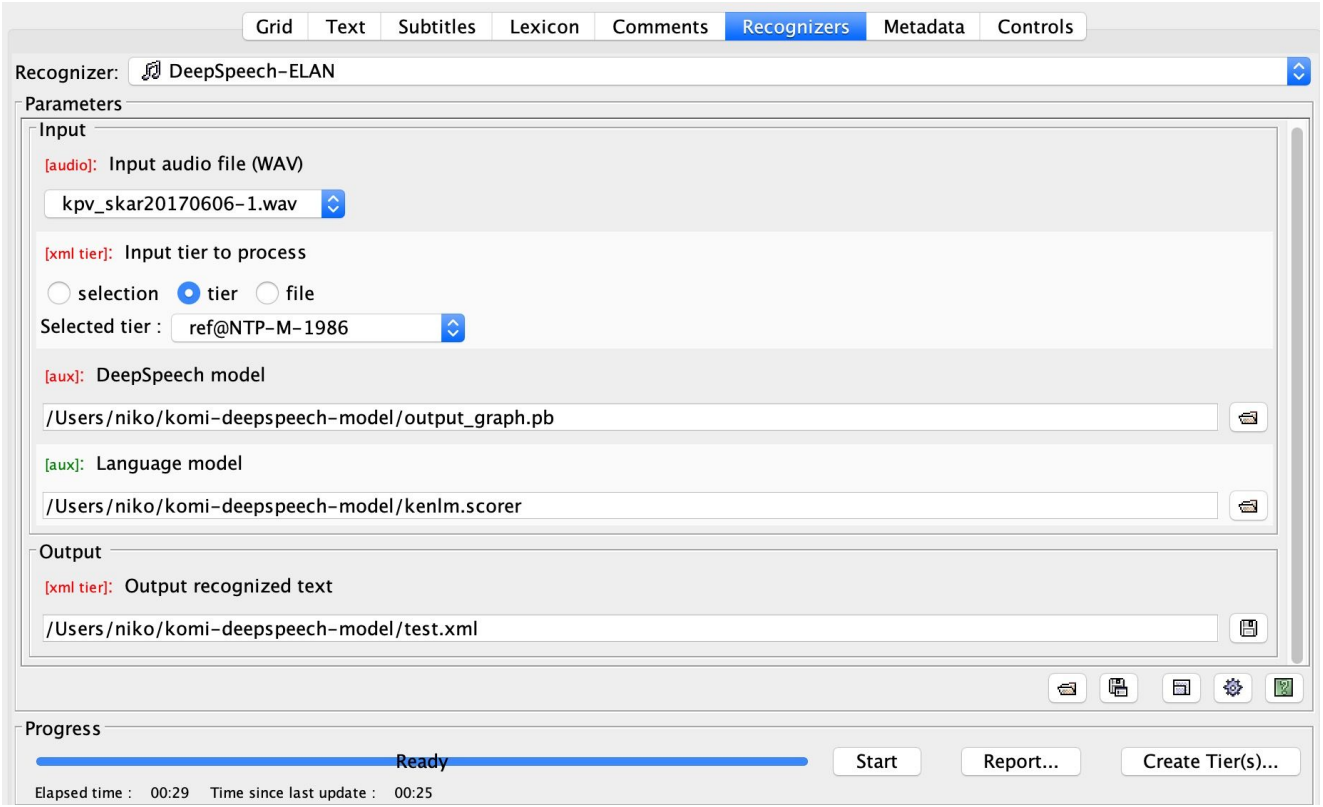
Figure 2: Results of our Nganasan and Kamas experiments compared with Wisniewski (2020)

Examples with different error rates

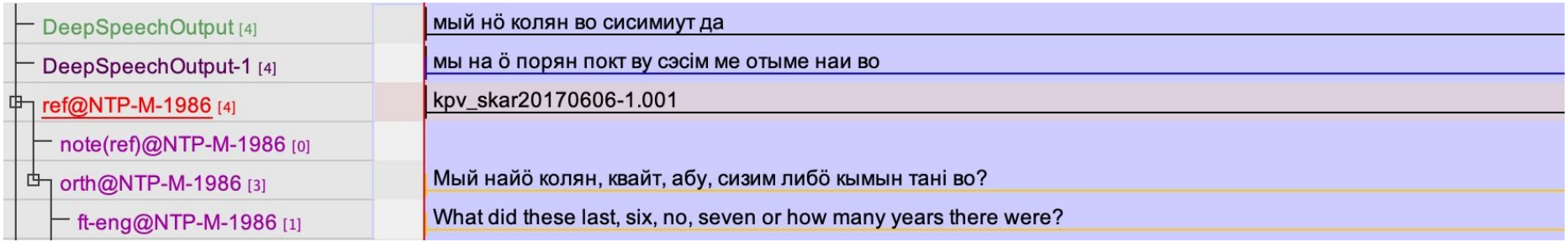
Ujabə ajirbi mĭnzərzittə.
1: ujabajrbimĭnzərzitəo
2: ujabaj irbi mĭnzərzitə
3: ujabajirbimınzirzitə
4: ujabajirbimınzərzitəo
‘He was reluctant to cook his meat.’

Mĭlleʔbi, mĭlleʔbi, ej kuʔpi.
1: mĭleʔtimĭleʔtiejkuʔpiö
2: müleʔpi mĭleʔtə ej kuʔpi
3: nuleʔbəmıleʔtəejkuʔpi
4: mıleʔpimıleʔpiejkuʔpia
‘He went, he went, he did not kill.’

ELAN DeepSpeech & ELAN Persephone plugins



ELAN DeepSpeech API & ELAN Persephone



How these systems are trained?

- Normally a collection of ELAN files and their audio is enough
- Transcriptions should be checked for unknown characters
- Normally segments longer than 10 seconds are ignored
- A script reads all ELAN files and creates audio clips from the segments
- We end up with a directory that contains small wav and text files:
    - sentence_0001.wav
    - sentence_0001.txt

- The original audio cannot be reconstructed from the model (we think)!