UNIVERSITÄT
DES
SAARLANDES

A Parallel Corpus: preprocessing, annotation, feature design and extraction

Corpus-based Translation Studies

Maria Kunilovskaya

HS Empirical Linguistics and Translatology
MA Translation Science and Technology

May 16, 2024

Outline

1. (Parallel) corpus building workflow
   - research design
2. Alignment
3. Annotation
   - general
4. Automatic annotation
   - hands-on UDpipe
5. Manual annotation
   - possible setups
     - categorial or ordinal labels
     - span annotation (ex. errors)
     - direct assessment
   - reliability and validity studies

Conceptualise

1. Ask a genuine question. Get curious.
2. Put forward a testable hypothesis (or question) with an expected outcome (if A, then B).
3. Think of a method to solve your task.

Plan and build a raw corpus

4. Identify source(s) of cross-lingual/multilingual[1] data.
5. Assess the quality and usability of data: What's the format?
6. Extract raw text.
7. Align at sentence level.
8. Pre-process: standardise, clean and filter, and balance the categories.
9. Structure the raw corpus: Decide on the unit of storage, add IDs, arrange the meta-data and collect descriptive parameters.

[1] What's the difference?

Parallel Corpus structure and storage

1. a wide table:
   columns=['seg_id', 'source_raw_text', 'target_raw_text'],
2. a long table:
   columns=['seg_id', 'text_type', 'mode', 'raw_text'],
3. an xml format
4. a folder of files (File=Document, Line=Segment format) with folder and filenames reflecting the corpus structure: e.g.
   proj/mst/data/conllu/train/europ/deen/de/ORG_WR_DE_EN_000001.conllu

NB! Segment naming hints for tables (seg_id):

- use doc_id
- use leading zeroes to avoid overmatching

e.g.
ORG_WR_EN_DE_000021:03

Add more information: tags or labels/scores

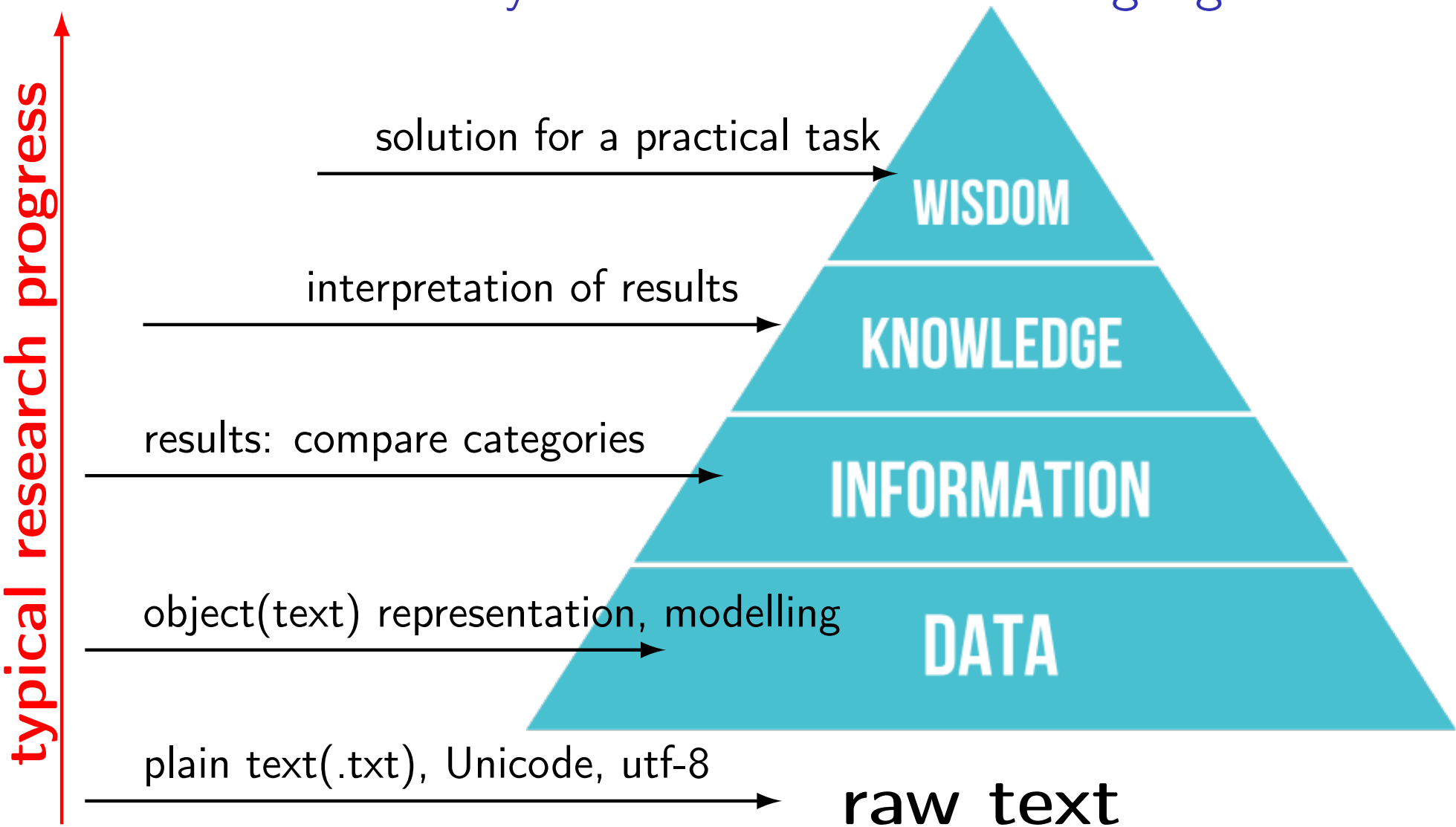
10. Do you need annotation?
    - automatic (tagging by a software)
    - manual (annotation/evaluation by people/experts)

Congrats! 80% of the research is done!

Numeric representation or extracting quantitative parameters

11. Design or select a meaningful way to convert language to numbers (aka modelling).
12. Extract and arrange the quantitative information (a data table).
13. Apply your main research method(s) and get the results.
14. Analyse and interpret the results (inc. evaluation)
    - what is your reference, baseline, expected level? or how do the categories in your data compare?
    - support or reject the hypothesis.
    - explain the results, run additional tests to support your speculations.
15. (Apply to a real world problem)

Information hierarchy in research based on language data



Parallel documents to sentence-aligned wide table

[148531] Empirical Linguistics and Translatology

```
python3 get_aligner_bash.py
    --indir /absolute/path/to/anno/data/raw/deen/

cd aligner/
    chmod a+x deen_europarl-uds.sh
    # for fresh LFAligner download on Linux:
    chmod +x scripts/*
    ./deen_europarl-uds.sh

python3 get_raw_aligned.py --indir
    data/raw/temp/deen/align_2024.05.16/ --lpair deen --docsize 0
    --cutoff 0.2
```

(Optionally) To report human evaluation results for the automatically aligned corpus:

- Get a random sample of documents from your wide table.
- Ask an annotator to correct the alignment in a TMX-Editor (e.g. https://github.com/heartsome/tmxeditor8) and
- Calculate the F1-score which shows how many segment pairs were aligned correctly:

```
python3 eval_alignment.py -t data/ -g data/ -d deen
```

Raw vs annotated

'The results are only as good as the corpus' (Sinclair, 1991)[6]

Research design:
'trust the text' vs resources with 'added value'

Types of 'added value'

Metadata and markup

objective information about the data and document structure

1. chronotope: time and location of production
2. speaker's gender, age, profession
3. document structure: headings, by-lines, paragraph breaks

Auto or manual annotation

interpretative information: reflects results of linguistic analysis

1. register or genre
2. topical domain
3. part-of-speech (PoS), syntactic structure
4. semantic categories
5. discourse phenomena
6. human judgment labels/scores

Markup and annotation level

What is annotated? What is the unit of annotation?

1. corpus storage unit as a whole
   - ex. documents or sentences for their source (MT system, education level of the speaker), sentiment, genre, translation quality (TQ), etc.
2. elements inside the corpus storage unit and their relations
   - ex. PoS, syntactic relations, anaphora, discourse relations, named entities, translation errors, etc.

Storage formats

How to store markup and annotation?

1. standoff annotation (separate, cross-referenced files)
   - ex. RU_25_1.txt, RU_25_1.head.txt, RU_25_1.ann
   - ex. aligned lists of filenames and scores, labels, meta-information lines
2. versions of the corpus: tagged, lemmatised
   - ex. one-word-per-line CoNLL-U format
   - ex. lempos representation: be_AUX work_NOUN suit_NOUN on_ADP the_DET way_NOUN out_ADV ?_PUNCT
3. text integrated with annotation (XML-like formats)
   - ex. BNC XML Edition

Summary

Annotation is enhancing the corpus with interpretative information using a *"combination of manual and automatic methods to add tags, codes, and documentation that identify textual and linguistic characteristics of the data"* (Rayson 2015)[5]

Aspects to consider:

- Is your data ready to be annotated?
- What exactly do you need to know about your language items?
- How are you going to extract the info from annotation for the analysis stage?

Universal Dependencies framework and UDpipe

processors='tokenize,mwt,pos,lemma,depparse'

Why use standard linguistic annotation?

- PoS: use morphological features, disambiguate lemmas
- lemmatisation: reduce data sparsity and facilitate lexical items extraction; reduce size of language resources (LM, embeddings)
- syntactic parsing: access syntactic phenomena, specific structural types of phrases, clauses, sentences

NB! Word embedding pipelines that use sub-word information assume only raw text as input (ex. fastText, contextualised embedding models)

UD: get lemmas, PoS, morph. features and dependences

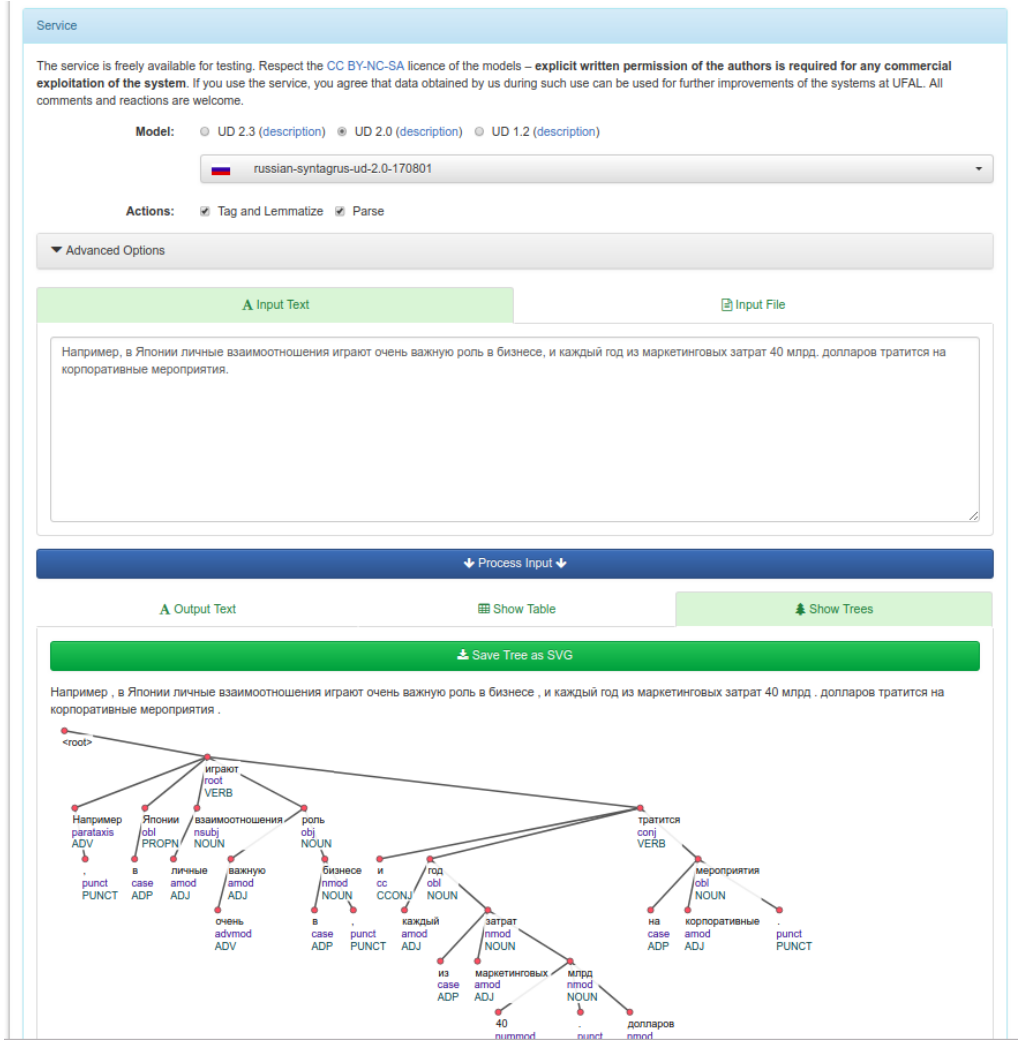
UD annotation setup for terminal (aka command line or cmd)

1. download udpipe-1.3.1-bin.zip from github,
2. extract the contents, get the executable binary file 'udpipe' from the folder for your OS and move it to a more convenient location (ex. ./ud/),
3. download the UD 2.5 models for your languages languages from lindat repo and put it in the same ./ud/ (ex. ./ud/german-gsd-ud-2.5-191206.udpipe)
4. run this command in the terminal from ./ud/ to test the setup:

```
echo "Du bist dran." | ud/udpipe -tokenize -tag -parse
ud/german-gsd-ud-2.5-191206.udpipe
```

On-line services for parsing and visualising syntactic trees

UDpipe



Quality of automatic annotation

see full quality evaluation report for the latest versions

Treebank	Mode	Words	Sents	UPOS	XPOS	UFeats	AllTags	Lemma	UAS
German-GSD	Raw text	99.6%	80.9%	91.7%	79.5%	69.8%	62.9%	95.4%	78.1%
English-EWT	Raw text	98.9%	77.4%	93.3%	92.8%	94.2%	91.3%	95.5%	80.2%

Excerpts from the performance tables for 2.5 models

.conllu format

Id	Form	Lemma	UPosTag	XPosTag	Feats	Head	DepRel	Deps	Misc
# newdoc									
# newpar									
# sent_id = 1									
# text = He was a middle-aged child that had never shed its baby fat, though some gifted tailor had almost succeeded in camouflaging his plump and spankable bottom.									
1	He	he	PRON	PRP	Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Prs	7	nsubj	_	_
2	was	be	AUX	VBD	Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin	7	cop	_	_
3	a	a	DET	DT	Definite=Ind|PronType=Art	7	det	_	_
4	middle	middle	ADJ	NN	Number=Sing	6	compound	_	SpaceAfter=No
5	-	-	PUNCT	HYPH	_	6	punct	_	SpaceAfter=No
6	aged	aged	ADJ	JJ	Degree=Pos	7	amod	_	_
7	child	child	NOUN	NN	Number=Sing	0	root	_	_
8	that	that	PRON	WDT	PronType=Rel	11	nsubj	_	_
9	had	have	AUX	VBD	Mood=Ind|Tense=Past|VerbForm=Fin	11	aux	_	_
10	never	never	ADV	RB	_	11	advmod	_	_
11	shed	sh	VERB	VBN	Tense=Past|VerbForm=Part	7	acl:relcl	_	_
12	it	it	PRON	PRP	Case=Nom|Gender=Neut|Number=Sing|Person=3|PronType=Prs	15	nmod:poss	_	SpaceAfter=No
13	s	be	PART	POS	_	12	case	_	_
14	baby	baby	NOUN	NN	Number=Sing	15	compound	_	_
15	fat	fat	NOUN	NN	Number=Sing	11	obj	_	SpaceAfter=No
16	,	,	PUNCT	,	_	7	punct	_	_
17	though	though	SCONJ	IN	_	23	mark	_	_
18	some	some	DET	DT	_	20	det	_	_
19	gifted	gift	VERB	VBN	Tense=Past|VerbForm=Part	20	amod	_	_
20	tailor	tailor	NOUN	NN	Number=Sing	23	nsubj	_	_
21	had	have	AUX	VBD	Mood=Ind|Tense=Past|VerbForm=Fin	23	aux	_	_
22	almost	almost	ADV	RB	_	23	advmod	_	_
23	succeeded	succeed	VERB	VBN	Tense=Past|VerbForm=Part	7	advcl	_	_
24	in	in	SCONJ	IN	_	25	mark	_	SpacesAfter=\n
25	camouflaging	camouflage	VERB	VBG	VerbForm=Ger	23	advcl	_	_
26	his	he	PRON	PRP$	Gender=Masc|Number=Sing|Person=3|Poss=Yes|PronType=Prs	27	nmod:poss	_	_

tab-separated fields:
own id, token, lemma, PoS
morph features, head id, relation

Annotate clean raw files in bulk

1. download a model for your language (ex. english-ewt-ud-2.5-191206.udpipe)
2. to install the Python bindings in terminal, run: `pip install ufal.udpipe`
3. go to the folder with the script and the clean texts and run:

```
python3 UDparser.py --input data/raw_structured/deen/de/
      --output data/conllu_structured/deen/de/ --model
      ud/german-gsd-ud-2.5-191206.udpipe
```

Expected outcome: a folder with files in CoNLL-U format (*.conllu)

Parse and lemmatise a structured parallel doc-aligned corpus

Setting up

run command: `python3 multiling2conllu2lempos.py --root data/raw_register_corpus/`
Requires: udpipe binary, Python bindings, UD2.5 models next to the script and corpus (with languages folders at last-level),

1. pass --lempos, if you want lempos along with conllu
2. inspect lempos cleaning/filtering in cleaners.py
3. adjust the languages names

Expected output

Two folders with the same corpus structure as the input with:

1. UD-parsed docs (*.conllu)
2. lemmatised and tagged docs (*.lempos)

Lemposed text:
`i_PRON be_AUX not_PART make_VERB this_DET up_NOUN ._PUNCT`

The script can be adjusted to produce lemmatised output:
`i be not make this up .`

Get frequencies from CoNLL-U format

1. get the subcorpora size after preprocessing and annotation (sentence and word counts)
   - `python3 wc_walks_UDfolders.py --root /path/to/folder/ --depth 2 --minlen 2`
2. get a table with texts in rows, metadata from the corpus structure and normalised frequencies for 29 UD-based features in columns (ex. *sentlength*, *ppron*, *possdet*, *indef*, *cconj*, *sconj*, *copula*)
   - the script expects rip_ud_trees.py, some_extractors.py and searchlists/ in the same folder
   - requires categories names for values in your corpus structure: ex. `--levels doc register type lang` for corpus_structure/popsci/target/ru/text_1.txt

```
python3 extract_feature_values.py
      --root data/conllu_register_corpus/
      --out res/data_table_translated_registers.tsv
      --langs en ru --levels doc register type lang
```

Aspects to consider

1. Data selection,
2. Nominal, ordinal or interval scale (type of outcome variable),
3. Desirable vs doable: annotation scheme and guidelines,
4. Human factor: requirements to annotators,
5. Annotation tools and environments,
6. Provisions for quality control,
7. Reliability and validity studies,
8. Rules for consolidation stage.

NB! Pilot studies help test the setup, calibrate the judgments, internalise the guidelines and prepare the workflow for reliability evaluation!

Produce a detailed experiment description!

'Choose-the-best-option' task: Genre annotation

Task: Read the text. Indicate the degree to which you agree with the 10 statements on a Likert scale (see guidelines)
Motivation: to validate the model for predicting text functions
Annotation interface: GoogleForms look-n-feel
Raters: 3 linguists with PhD degrees
Data: 70 texts, random selection from unseen corpora

Setting up: general concept, adding questions in bulk by populating a template from FormCreator Add-on (in your GDrive: Add-ons>FormCreator>Create Form)
Raw Results table: have a look here

Reliability study: Krippendorff's $\alpha > 0.537$ online calculator
Consolidation stage: In cases of triple disagreements (48 items out of 700), the values were averaged and rounded to the closest score.

Another GoogleForms setup: Compare two translations

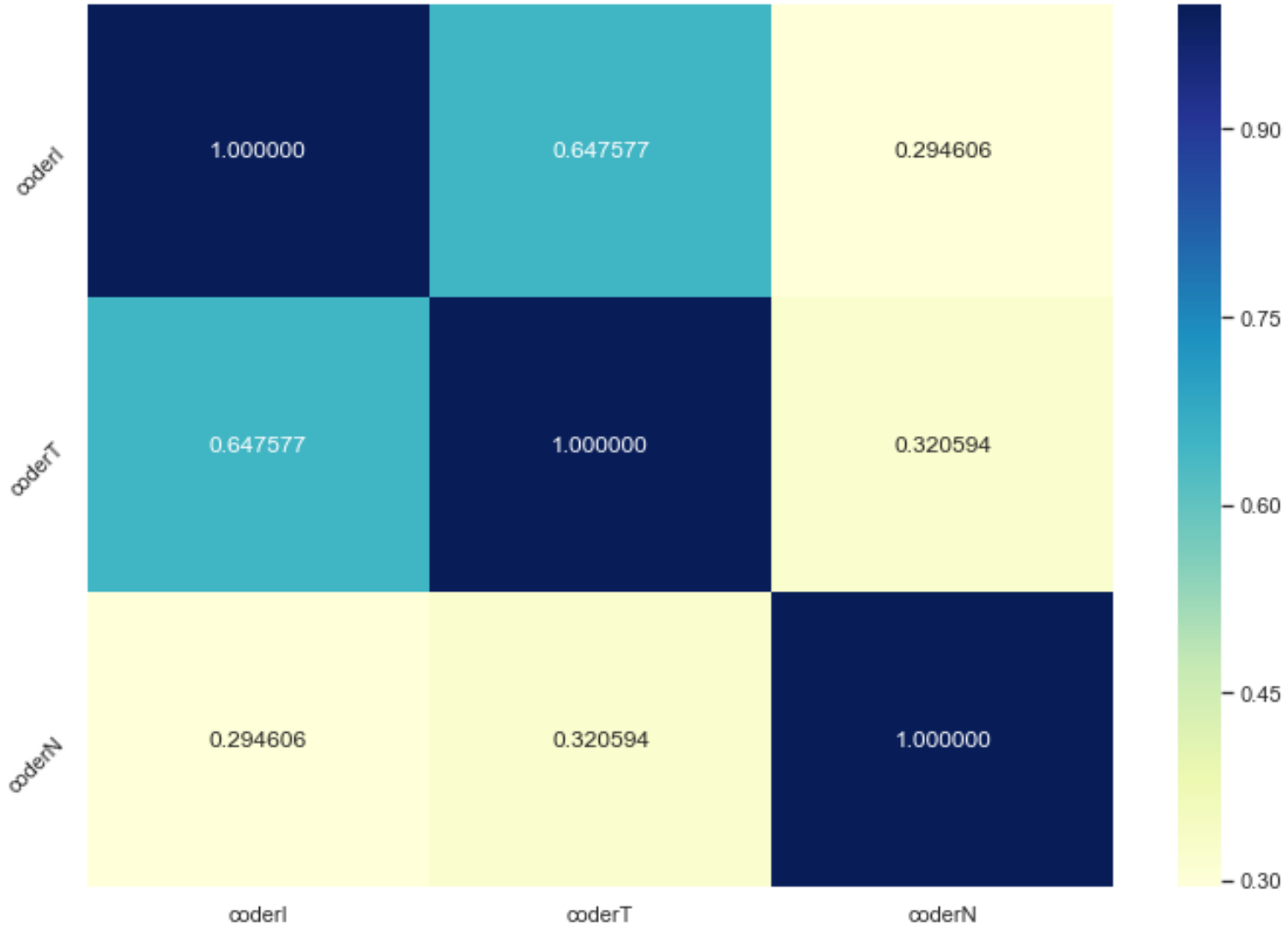
Task: Read the source text and assign each translation to 'good', 'bad' or 'same' category
Motivation: to verify the binary TQ labels from error annotation
Raters: 3 linguists with PhD degrees

Scale: nominal labels
Annotation setup: Input table, GoogleForms task
Reliability study:

1. total agreement (out of 80): 42, 52.5%
2. three different values: 6, 7.5%
3. unweighted average pairwise Cohen's $\kappa = 0.42$
4. weighted Krippendorff's $\alpha > 0.52$

Consolidation stage: items with triple disagreement filtered out of the initial dataset
Processing output: see commented **reliability.py** script

Heatmap for pairwise Cohens kappa



A dedicated annotation tool for translation errors

brat rapid annotation tool

NB! requires installation on a server and skills accessing the server

The online annotation environment used for grading and individual feedback in a practical translation course for 6 years

https://dev.rus-ltc.org/brat/#/rusltc/RU_1_146/RU_1_146_4

Data in the stand-alone annotation files is used for TQ research

Problems:

- demonstrating reliability for spans location and the categorisation of spans;
- maintaining a multi-parallel TMX;
- linking the spans annotated as translation errors to the SL items to identify the "areas of the learning curriculum where teaching is most needed"[2]

Annotating in a spreadsheet

Task: Given a source sentence and a set of 'bad' targets along with the error statistics for them,

1. highlight the spans in the source that were likely to trigger most of the errors;
2. categorise the span, choosing from a 40-item list of 'usual suspects' from the textbooks.

Data: 5 most challenging source sentences, calculated from errors in at least 6 translations; $\approx$ 500 target sentences per annotator

Spreadsheet: have a look

Reliability study: based on manual alignment of all selected items in the cross-annotated sample of 30 source sentences

span location same location: 81% of the average pairwise number of spans

categorisation weighted Kendall $\tau > 0.76$ (a correlation statistic for the ranked data in the range of (-1, 1), allows for ties)

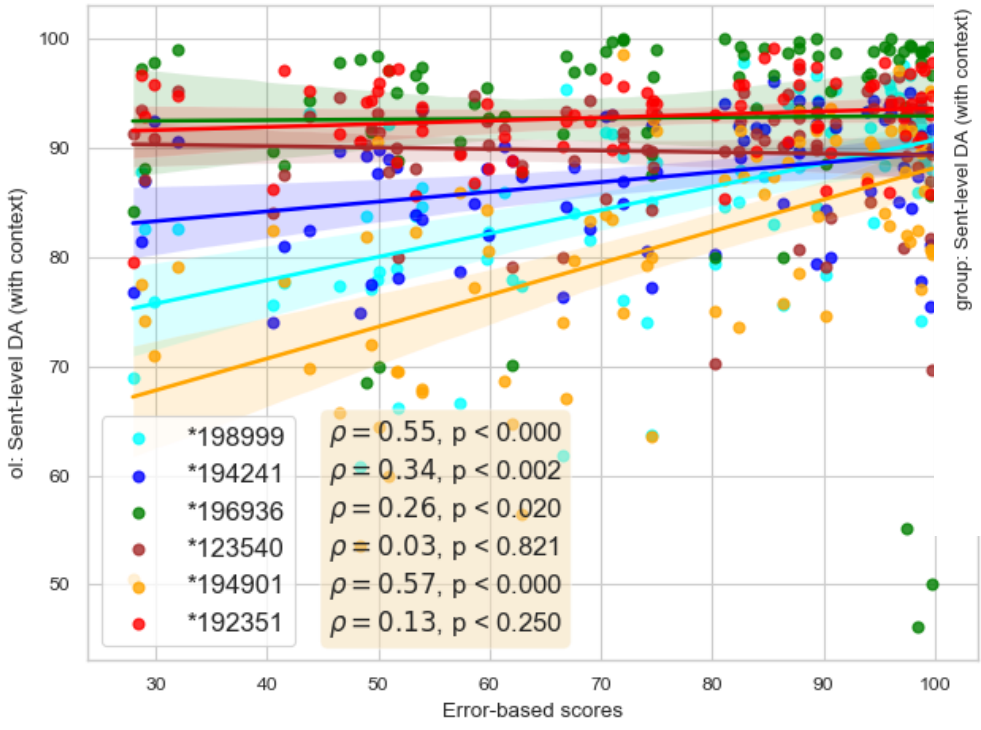
QuestionPro platform for DA

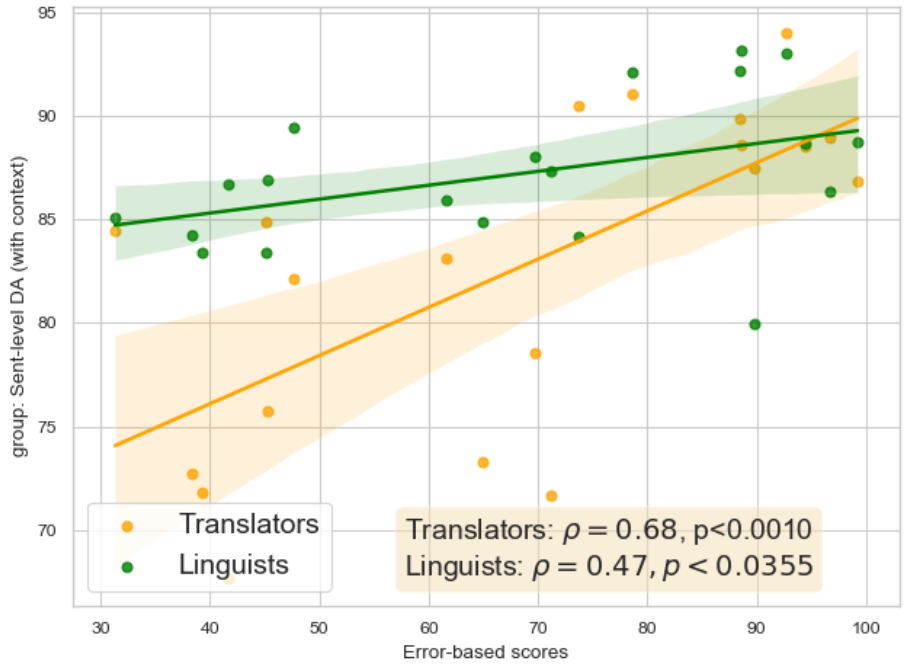
Task: Read the source text. Use the slider to indicate how much you agree that the text in bold is an adequate translation of the original English sentence, given the context.
Motivation: correlate human perception and scores from error annotation
Raters: 12 final year BA students (linguistics, translation)
Data: 150 targets to 40 ST with 6 scores for each sentence in context

Implementation:
https://www.questionpro.com/a/listSurveys.do

Reliability: Ratio of items with triple disagreement (30 pts diff in scores): 29.2%(translators); 6.3% (linguists)
Validity: correlation with error-based scores

Spearman correlation with 'gold' scores from error analysis





Overviews of manual annotation setups

Comprehensive discussions of designs for manual annotation experiments, given from the reliability study perspective:

1. The Unified and Holistic Method Gamma (γ) for Inter-Annotator Agreement Measure and Alignment[4]
2. Inter-Coder Agreement for Computational Linguistics[1]
3. Assessing Inter-Annotator Agreement for Translation Error Annotation[3]

See dedicated scripts in the Implementation section

References I

[1] Artstein, R. and Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. Computational Linguistics, 34(4):555–596.

[2] Castagnoli, S. (2011). Exploring variation and regularities in translation with multiple translation corpora. Rassegna Italiana di Linguistica Applicata, (43(1)):311–332.

[3] Lommel, A., Popović, M., and Burchardt, A. (2014). Assessing Inter-Annotator Agreement for Translation Error Annotation. In Language Resources and Evaluation, pages 31–37.

[4] Mathet, Y., Widlöcher, A., and Métivier, J.-P. (2015). The Unified and Holistic Method Gamma (γ) for Inter-Annotator Agreement Measure and Alignment. Computational linguistics, 41(3):437–479.

References II

[5] Rayson, P. (2015). Computational tools and methods for corpus compilation and analysis. In Biber, D. and Reppen, R., editors, Cambridge Handbook of English Corpus Linguistics. Cambridge University Press.

[6] Sinclair, J. (1991). Corpus, Concordance, Collocation. Oxford University Press.