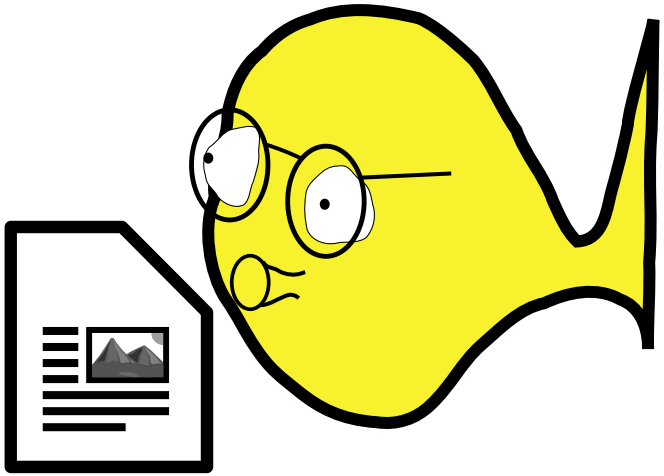


Adding Neural Machine Translation to LibreOffice

From Model to Product

Thomas Viehmann, MathInf GmbH
tv@mathinf.eu

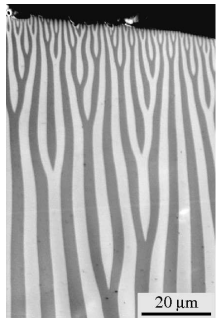
Hacking Machine Learning, 8 November 2018

About me

Thomas Viehmann (@tom on PyTorch, @t-vi on Github)

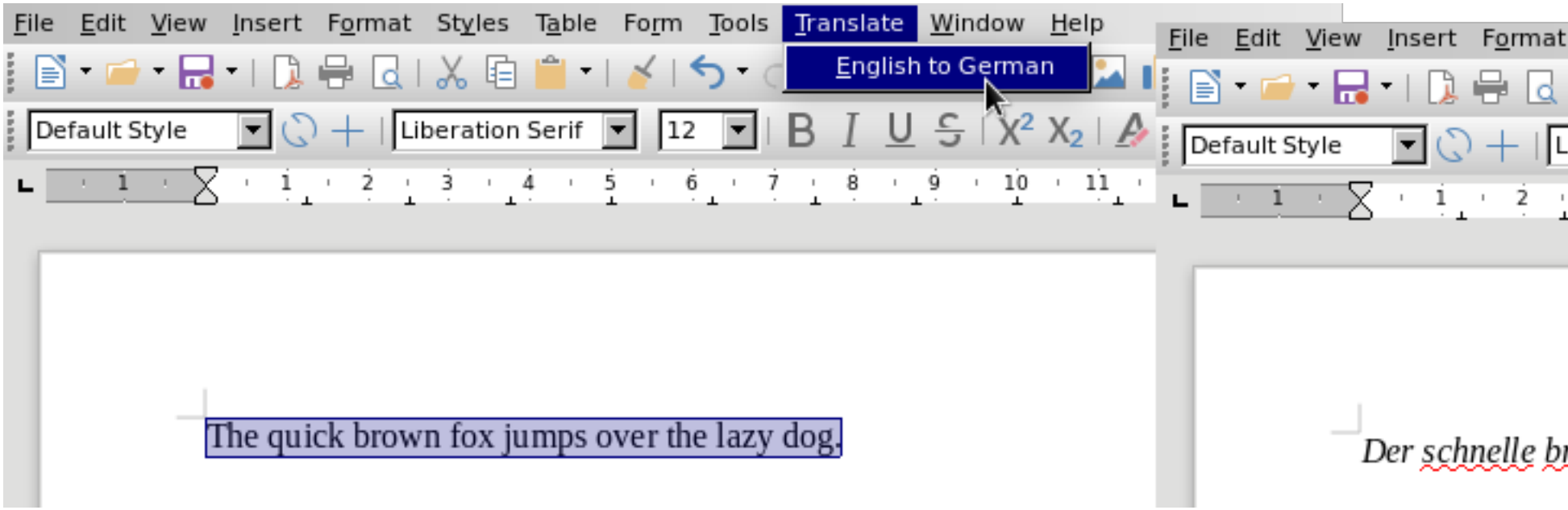
- core PyTorch developer – contributed some 150 features and bugfixes to PyTorch
- specialist ML and PyTorch training and consultancy MathInf GmbH

- ML blog: `https://lernapparat.de/`
- Background: Ph.D. in Mathematics (Bonn)
  Mathematical modelling of fractal behaviour in ferromagnets



LibreOffice Translate

Idea: 1-click translation of text with formatting.



`https://github.com/lernapparat/lotranslate/`

Initial Development supported by the German Ministry of Research and Technology through the Prototype Fund. (Thank you!)

Prototype Fund

SPONSORED BY THE
Federal Ministry of Education and Research

Why not just use Google Translate?

- Privacy! Both my own and possibly restrictions for data that is not my own.
- Comfort: If I have a document, I will lose formatting by copying and pasting.
- You might be offline. (?)
- Customization: We can improve the models / create our own models for our favourite language pairs.
  - The LibreOffice localization community is awesome, so enable them to create models.
  - Occasionally, there are people with models looking for front-ends.

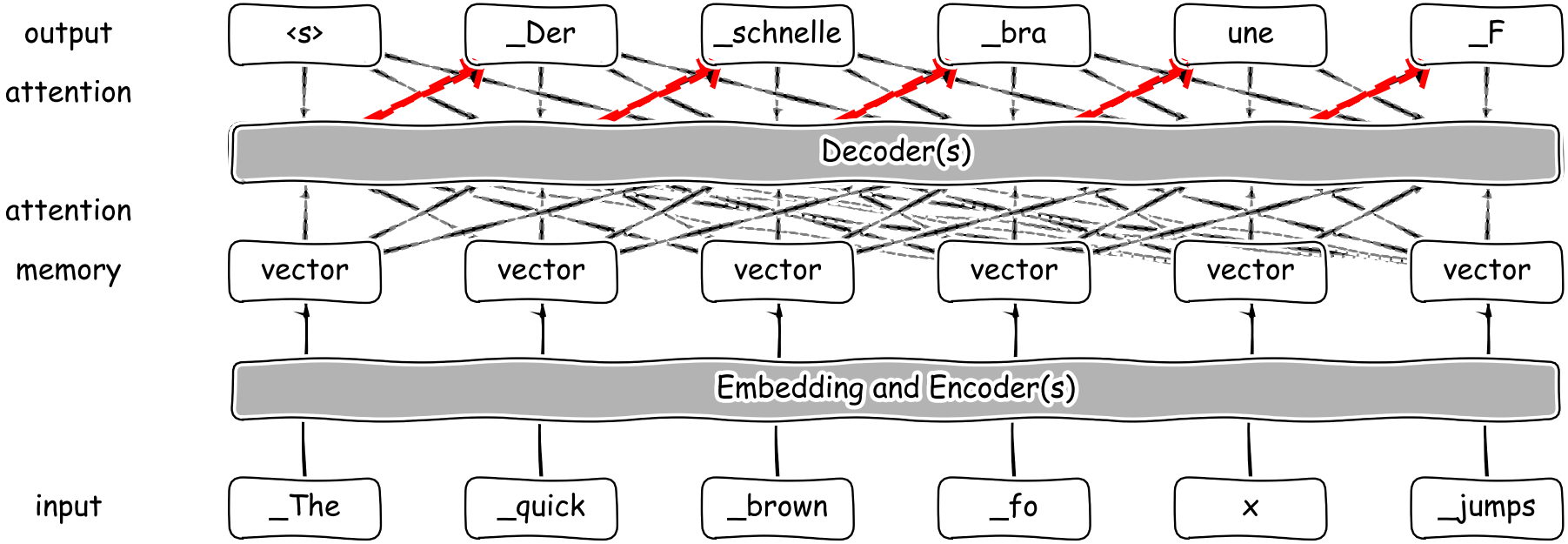
...because it is more fun to hack you own!

Not reinventing the wheel

We will use

- LibreOffice Writer – we will do a plugin, so we don't need to parse files, do UI, ... etc. Also helps with reaching users and model owners.
- OpenNMT – a Neural Machine Translation library. It comes in two flavours (PyTorch and Tensorflow), I am biased towards the PyTorch one. They do have training examples etc.
- Sentencepiece – a library developed by Google for "subword" tokenization (more in a bit).

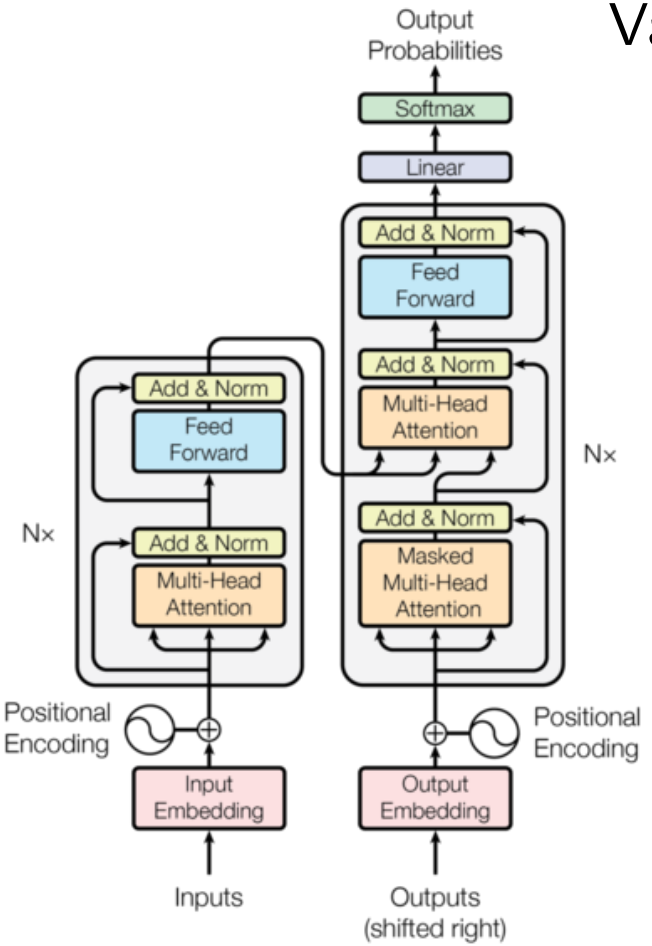
Oversimplified Neural Machine Translation Model



"Flagship" models: Encoder / Decoder based on Transformer. (The same architecture as BERT etc.)

Red arrows: Predictions. Attention full in Encoder and "causal" in Decoder.

Transformers in more detail...



Source: Vaswani et al.

Vaswani et al.: Attention is all you need

- Replace recurrence with attention
- Compute Keys along with Values (=Outputs of Encoder/Lower layers)
- Weight Keys with Query:

$$Att(Q, K, V) = softmax(\frac{QK^T}{\sqrt{dim(K)}})V$$

- Mask ($K = -\infty$) for causality in decoder.
- Layernorm, residual connections, dropout.
- Multihead: concatenate several *Attn*s

Recommended version for reading: S.Rush: The Annotated Transformer

Subword splitting

- One of the problems in using word vectors & co also shows up here: UNK, the not-in-the-dictionary word. For example *Traktorreparaturwerkstatt*.
- For this reason, people use "byte pair encoding" for translation (Senrich et. al.: Neural Machine Translation of Rare Words with Subword Units). Idea:
  - Take alphabet as tokens.
  - Iteratively: combine the most frequent tokens to a new "combined token".
  - Stop if you have a target vocabulary size.

  A slight complication is that tokenization isn't unique anymore.
- Practical implementations: subword-nmt or SentencePiece.
  $\rightarrow$ No more UNK! (unless you use funny unicode)

Are we done?

Let's try translations:

The quick brown fox jumps over the lazy dog.

$\rightarrow$

Are we done?

Let's try translations:
The quick brown fox jumps over the lazy dog.
$\rightarrow$ Der schnelle braune Fuchs springt über den faulen Hund.

Are we done?

Let's try translations:
The quick brown fox jumps over the lazy dog.
$\rightarrow$ Der schnelle braune Fuchs springt über den faulen Hund.
The quick brown fox jumps over the lazy dog. The dog walked to Peter.
$\rightarrow$

Are we done?

Let's try translations:
The quick brown fox jumps over the lazy dog.
$\rightarrow$ Der schnelle braune Fuchs springt über den faulen Hund.
The quick brown fox jumps over the lazy dog. The dog walked to Peter.
$\rightarrow$ Der schnelle braune Fuchs springt über den faulen Hund.

Hm.

Are we done?

Let's try translations:
The quick brown fox jumps over the lazy dog.
→ Der schnelle braune Fuchs springt über den faulen Hund.
The quick brown fox jumps over the lazy dog. The dog walked to Peter.
→ Der schnelle braune Fuchs springt über den faulen Hund.

Hm. The model only translates one sentence at a time. **We need to split by sentence.**

spaCy can do that! And does it well!
...first choice, but has many dependencies, which is a headache for our extension.
(almost) dependency-free alternative: SynTok (heurisic sentence splitter)

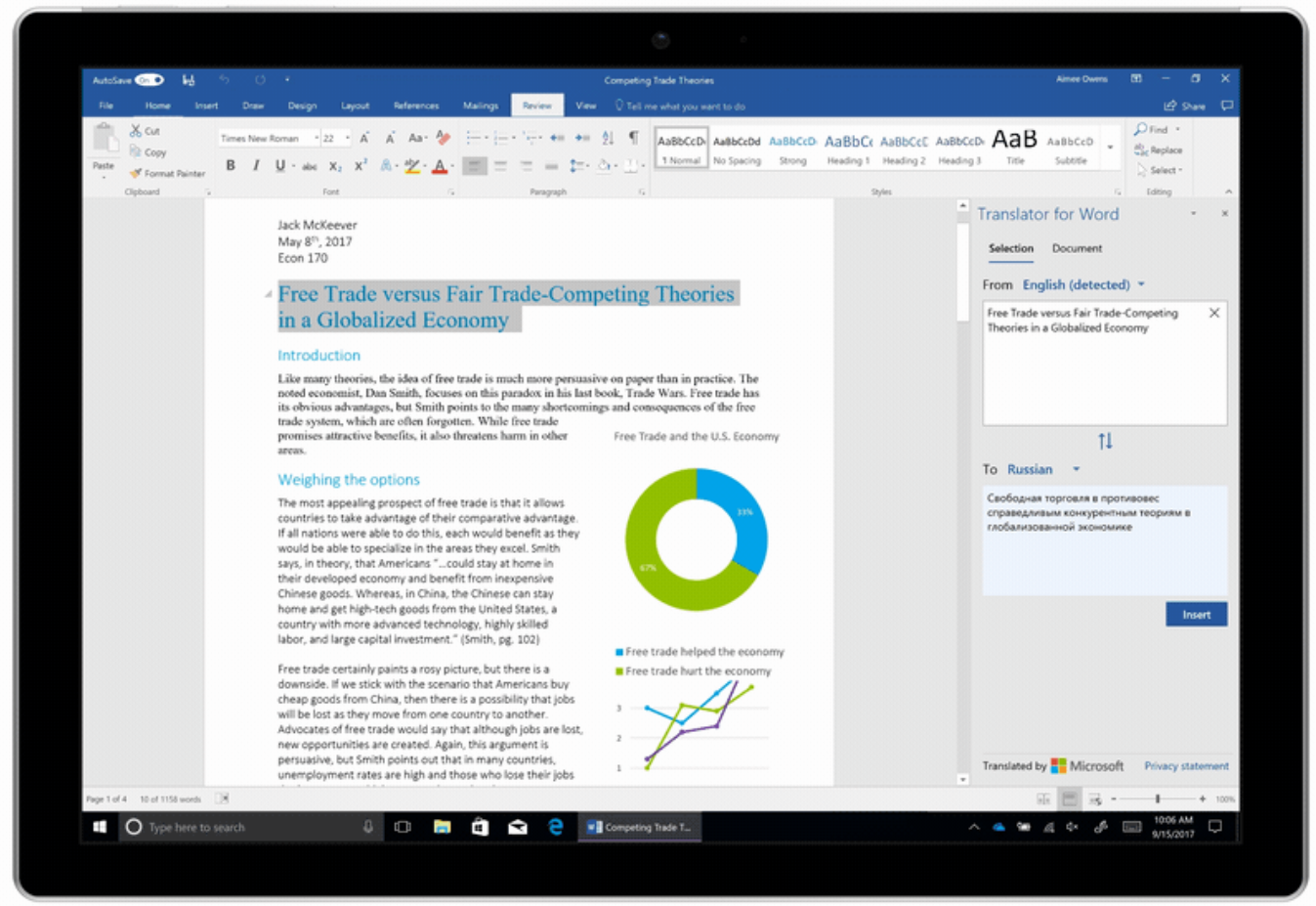
How to get formats? Attention!

	_The	_quick	_brown	_fo	x	_jump	s	_over	_the	_la	zy	_dog	.
_Der	0.07	0.42	0.13	0.03	0.01	0.03	0.02	0.02	0.01	0.02	0.01	0.00	0.24
_schnelle	0.02	0.76	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.01
_bra	0.00	0.03	0.90	0.02	0.03	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01
une	0.00	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.94
_F	0.00	0.00	0.06	0.02	0.82	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
uch	0.00	0.00	0.00	0.75	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.04
s	0.01	0.01	0.01	0.02	0.05	0.02	0.01	0.01	0.00	0.00	0.01	0.01	0.85
_spring	0.00	0.00	0.00	0.00	0.00	0.86	0.08	0.03	0.00	0.00	0.00	0.00	0.02
t	0.00	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.93
_über	0.00	0.03	0.02	0.00	0.00	0.04	0.03	0.03	0.04	0.11	0.02	0.01	0.67
_den	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.05	0.32	0.19	0.01	0.40
_fa	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.08	0.84	0.00	0.03
ul	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.06	0.73	0.00	0.20
en	0.00	0.01	0.03	0.01	0.00	0.06	0.01	0.03	0.01	0.04	0.03	0.06	0.71
_Hund	0.00	0.01	0.01	0.00	0.02	0.01	0.00	0.02	0.01	0.02	0.08	0.04	0.77
.	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.01	0.00	0.01	0.00	0.01	0.93
</s>	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.95

- use attention to map to match translated text to source words and use source formatting
- heuristics for “end-bias”
- process paragraph by paragraph
- attribution more accurate at expense of increased computation

How should a UI for machine translation look like?

A popular word processor uses the side bar:

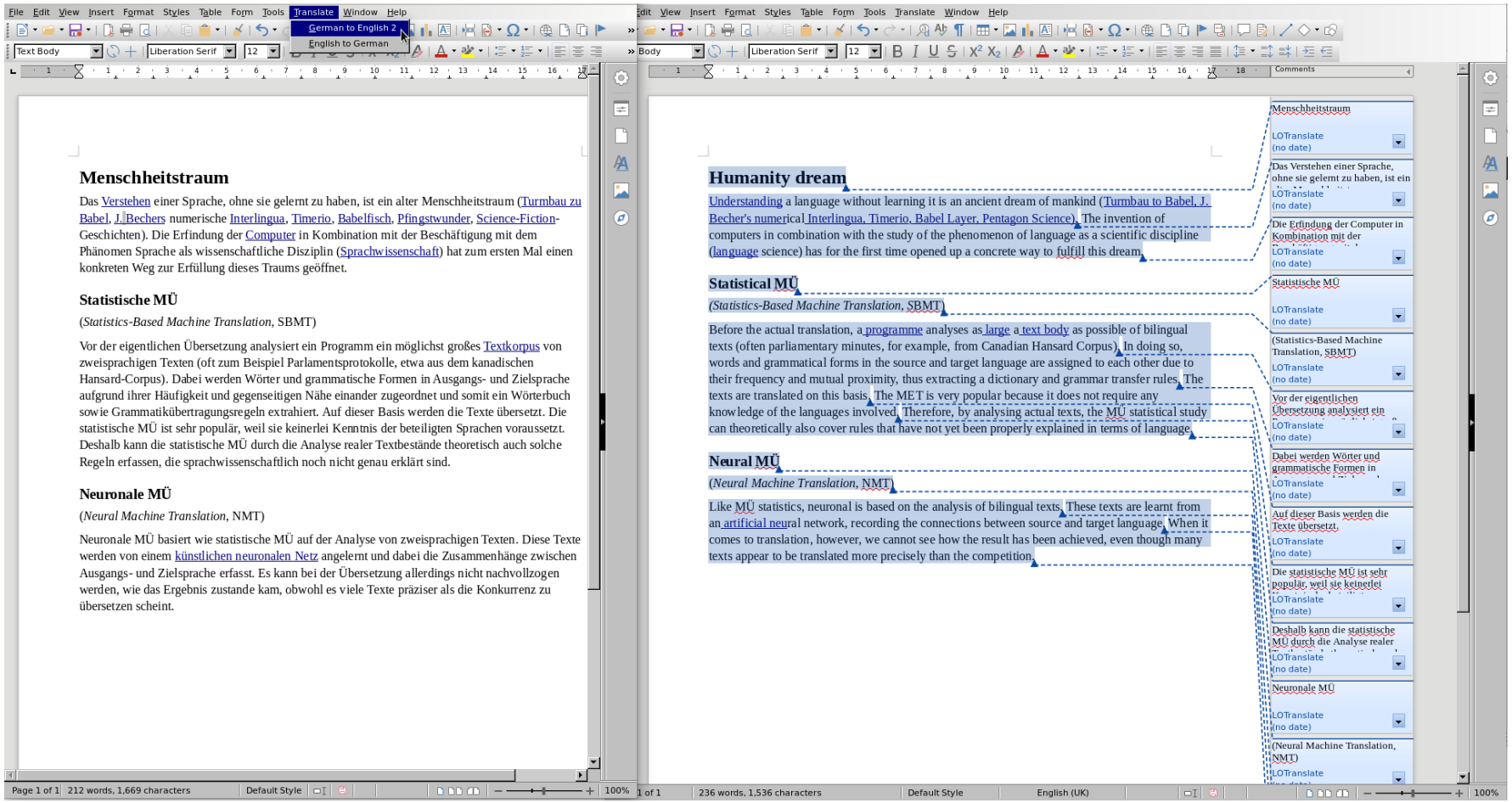


Source: Microsoft

Why?

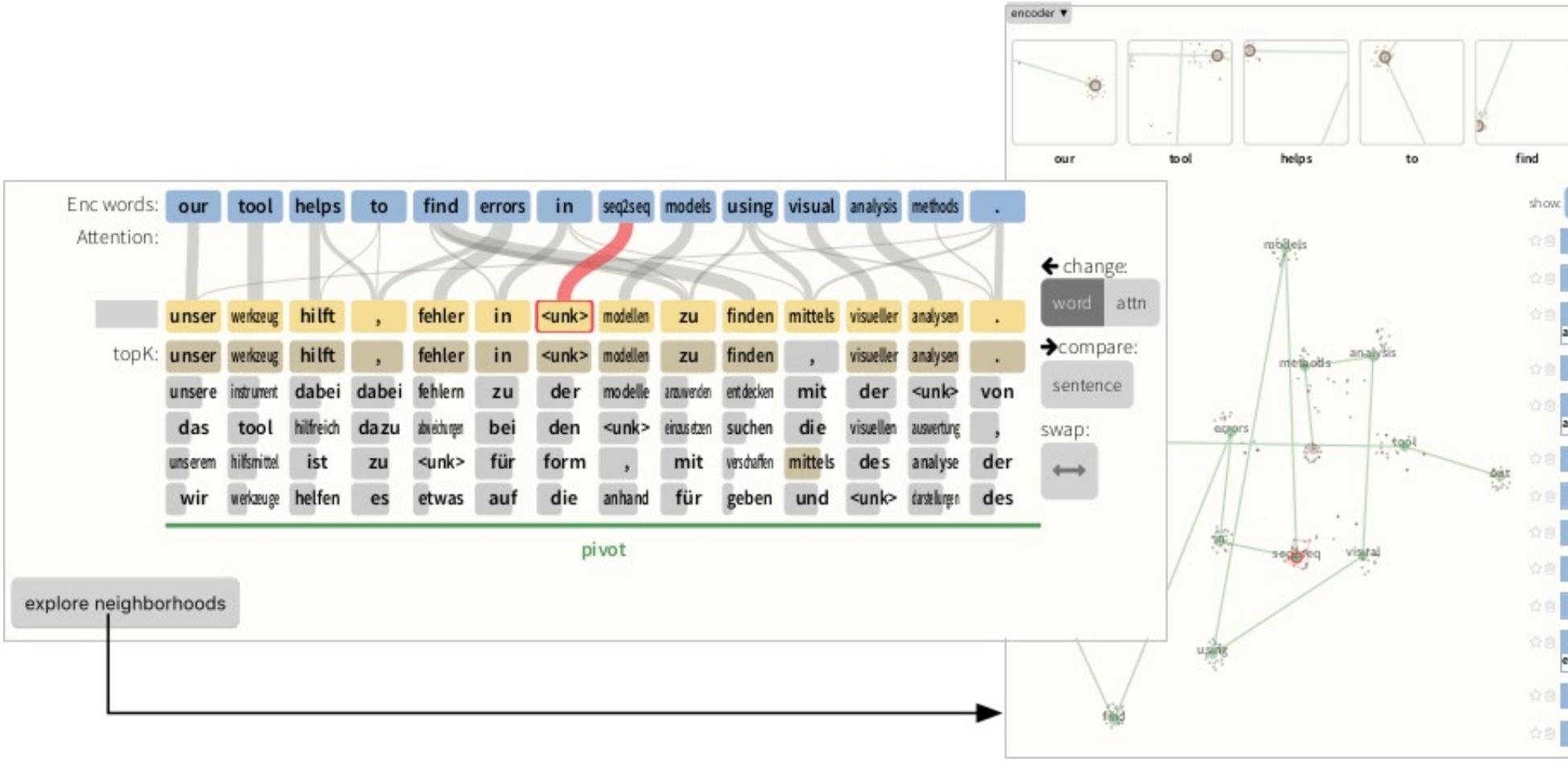
How should a UI for machine translation look like?

You might as well edit in the document (that's what word processors do!) rather than on the sidebar.
To be able to look at the old text if something is funny, use annotations.



How much UI do we need?

There are interesting visualizations of the model innards.



Source: Strobelt et al: Seq2Seq-Vis

...but what are the intervention options that make it useful?

Training new models

Basically we can follow the the OpenNMT tutorials

Requirements

- Needs parallel corpus of sentences (no word alignment needed) EN-DE: 4.5 Million pairs
- Needs (at least one) GTX1080Ti – for 1-2 weeks (that is 30-70 KWh per model – compare to 2.400 KWh per year for a family of 5 - and the GPU)

Steps

- Vocabulary preparation
- Training (this is what takes long)
- Evaluation (mostly "closeness" on a holdout set + inspection)
- Probably also want domain adaptation (i.e. specialize from "general" model to one specific for a domain, e.g. legal text).

Probably want a script / detailed tutorial...

Improvements: Modelling

- **Optimization**: OpenNMT has released CTranslate2, featuring a 2.5x speed increase on the CPU (4 cores) and 4x speed increase on the GPU.
  ...but as stern optimizers can we fix native PyTorch, too?
- **Model Distillation** from Transformer to RNN
  Senellart et al: OpenNMT System Description for WNMT 2018: 800 words/sec on a single-core CPU
  "Simple RNN to produce the intermediates of a trained Transformer"
  Or use DistilBERT like approach...
- Using **BERT as encoder** initialization of the seems to work reasonably well (Clinchant et al: On the use of BERT for Neural Machine Translation)
  computational expense, availability for many languages?
- **Faster training**

Improvements: Data

Data is a challenge for many language (pairs)

- **Open Corpora**: http://opus.nlpl.eu/
  Quality for various languages? (incidentally uses translated UI strings as one source)
- European Legislation
- Can we use word-by-word **dictionaries**, too?
  We have some of those.
- Can we use **weakly aligned texts**?

  e.g. gutenberg.org, LibreOffice documentation

  Related research:
  Automatic filtering of noisy corpora (WMT challenge 18 & 19)

LibreOffice

- Awesome product as a user.
- Great and friendly people developing it.
- The extension API can be quite unwieldly (a lot of RPC-style things going on) and hard to debug...
- ...but there is a Python binding (yay!)

Complications in the build process

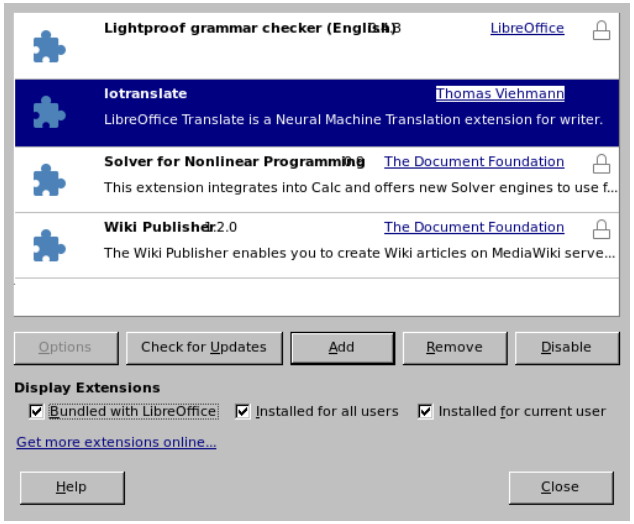
- We need a few Python modules as dependencies: OpenNMT, PyTorch, SynTok (to get sentences) + quite a few indirect ones
- Some are platform / Python-Version specific
- On Windows, LibreOffice ships it's own Python; on Debian it uses the system one, ...
- Current solution: build "simple" OXT
- For Windows (only) ship OXT with all dependencies:
  - Install dependencies in fresh LibreOffice with pip
  - Copy site-packages into OXT zipfile (30MB $\rightarrow$ 150MB)
- Currently using Exclipse, but maybe want to do something more automated...

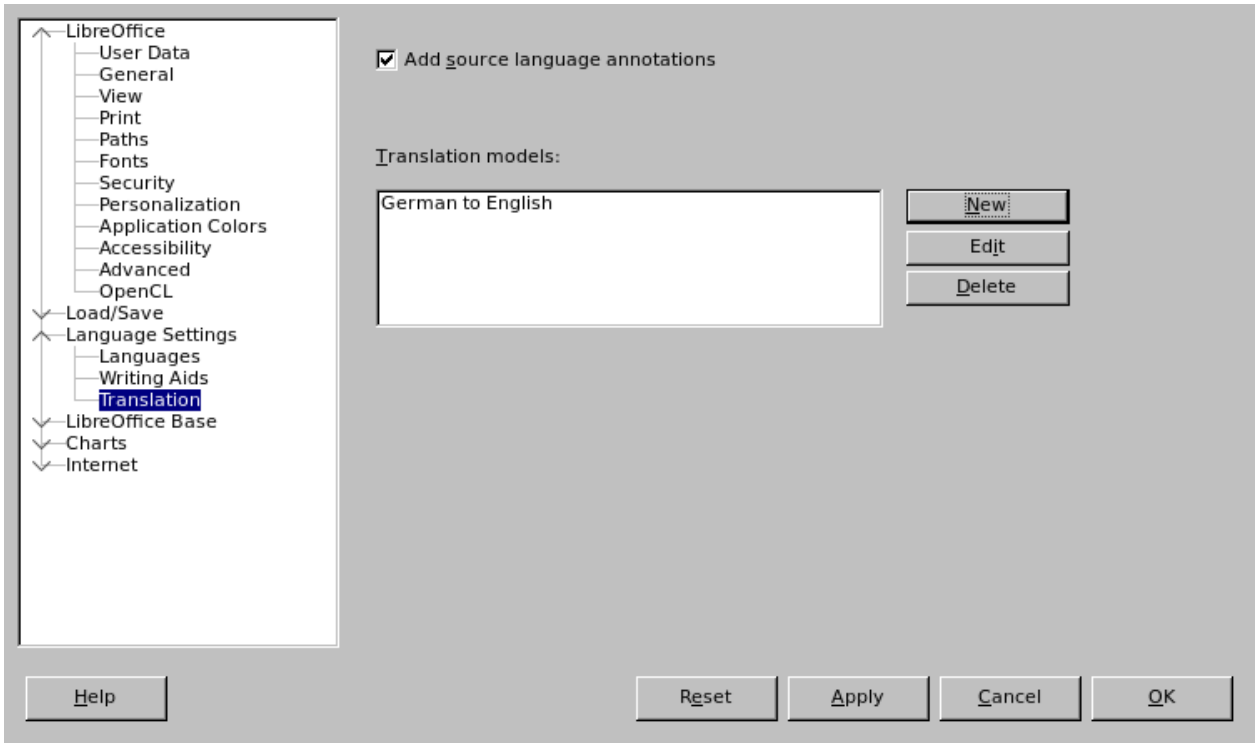
$\rightarrow$ Probably want to shed some dependencies, maybe port to C++.

We might dream of becoming part of LibreOffice proper...

Installation

First install the extension.





Download the model, unzip it and install it in the preferences.

Summary

- Open source Neural Machine Translation is advanced enough to be useful. Yet, exiting improvements to be had are all around.
- Getting it *ready for the user* takes some work.
- Real chance to get community-developed models if we make training and modelling easy.

If you want to hack on LibreOffice Translate, do give me a shout!

Thank you!

Your questions and comments

brought to you by
MathInf GmbH – Speciality PyTorch Training and Consulting
Contact: Thomas Viehmann, tv@mathinf.eu
Code and slides at
`https://lernapparat.de/libreoffice-translate/`