

Teaching the Technical Dimension of Neural Machine Translation Using Python-Based Jupyter Notebooks

Colloque interdisciplinaire « vers une robotique de traduire? »

Université de Strasbourg, 30 septembre – 1er octobre 2021

Professor Ralph Krüger

Institute of Translation and Multilingual Communication

TH Köln, Germany

Prelude

Machine Translation Literacy

“Basically, MT literacy means knowing how MT works, how it can be useful in a particular context, and what the implications are of using MT for specific communicative needs. [...] Knowledge of MT is a form of digital literacy.”

(O’Brien/Ehrensberger-Dow 2020:146-147)

MT Competences in the EMT Competence Framework



This competence³ includes all the knowledge and skills used to implement translation technologies. It also includes basic knowledge of machine translation technologies and the ability to implement machine translation according to potential needs.

STUDENTS KNOW HOW TO...

Skill descriptor

- 15 Use the most relevant IT applications, including the full range of office software, and become familiar with new IT resources
- 16 Make use of search engines, corpus-based tools and CAT tools as well as CATA (computer aided text analysis) tools
- 17 Pre-process, process and manage files and other media/sources as part of the translation, e.g. video and multimedia files, handle web technologies
- 18 Master the basics of MT and its impact on the translation process
- 19 Assess the relevance of MT systems in a translation workflow and implement the appropriate MT system where relevant
- 20 Apply other tools in support of language and translation technology, such as workflow management software

(EMT 2017:9)

The Literate Computing Paradigm

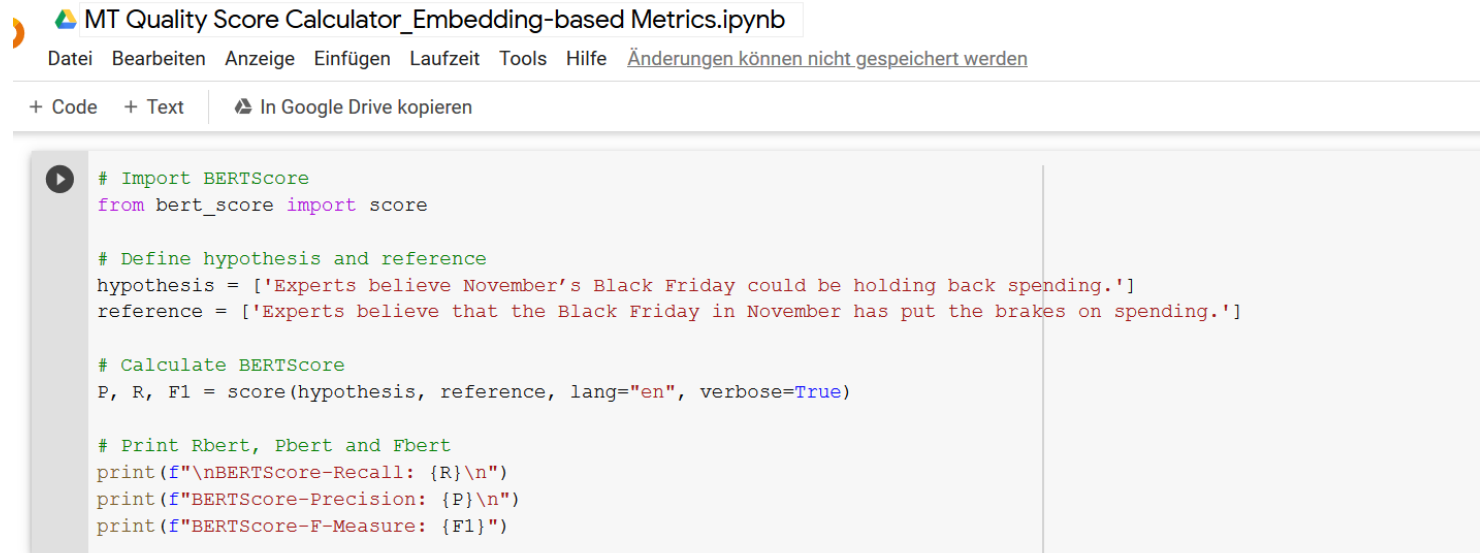
“A literate computing environment is one that allows users not only to execute commands but also to store in a literate document format the results of these commands along with figures and free-form text that can include formatted mathematical expressions. In practice it can be seen as a blend of a command-line environment such as the Unix shell with a word processor, since the resulting documents can be read like text, but contain blocks of code that were executed by the underlying computational system.”

(Millman and Pérez 2014:20, boldface added)

Jupyter Notebooks

Definition and example notebook

- interactive, web-based computing resources for developing, documenting, and executing code, as well as communicating the results (Jupyter Team 20215)



The screenshot shows a Jupyter Notebook interface. At the top, there's a title bar with a file icon and the text "MT Quality Score Calculator_Embedding-based Metrics.ipynb". Below the title bar, there's a menu bar with options: "Datei", "Bearbeiten", "Anzeige", "Einfügen", "Laufzeit", "Tools", "Hilfe", and a status message "Änderungen können nicht gespeichert werden". Below the menu bar, there's a toolbar with icons for "+ Code", "+ Text", and "In Google Drive kopieren". The main area of the notebook contains a code cell with the following Python code:

```
# Import BERTScore
from bert_score import score

# Define hypothesis and reference
hypothesis = ['Experts believe November's Black Friday could be holding back spending.']
reference = ['Experts believe that the Black Friday in November has put the brakes on spending.']

# Calculate BERTScore
P, R, F1 = score(hypothesis, reference, lang="en", verbose=True)

# Print Rbert, Pbert and Fbert
print(f"\nBERTScore-Recall: {R}\n")
print(f"BERTScore-Precision: {P}\n")
print(f"BERTScore-F-Measure: {F1}")
```

R_{BERT} , P_{BERT} and F_{BERT} are given as tensors, each containing a scalar value. The tensor for F_{BERT} (BERTScore-F-Measure in the output above) is the final BERTScore for this hypothesis-reference pair. BERTScore rescales this output so that it lies in a range between 0 and 1, with 0 being a very poor and 1 being a very good score (so, BERTScore is a similarity measure, just as the traditional F-Measure). The hypothesis-reference pair defined above is scored very highly because the two sentences express almost the same meaning, although they differ in their respective surface forms. Due to these formal differences between hypothesis and reference, traditional MT quality scores such as F-Measure and BLEU would score them much lower than BERTScore does.

(https://colab.research.google.com/drive/1pK479-xhDSDHrJA4OTfU1tXV6Pwp_G4f?usp=sharing#scrollTo=Y1oBerglbfsU)

Jupyter Notebooks

Raw notebook document

- notebooks are written in [Markdown](#):

2.1.2 BLEU

BLEU is an acronym for **Bilingual Evaluation Understudy**. It was originally proposed in [Papineni et al. (2002): BLEU: A Method for Automatic Evaluation of Machine Translation](<https://www.aclweb.org/anthology/P02-1040.pdf>). Today, it is still the most commonly used automatic MT quality score and it achieves reasonably good correlation with human quality judgements. BLEU is calculated using the following formula:

$$\text{BLEU} = \min(1, \frac{\text{hypothesis_length}}{\text{reference_length}}) \left(\prod_{i=1}^n \text{precision}_i \right)^{\frac{1}{n}}$$

Let's break this down:

$\left(\prod_{i=1}^n \text{precision}_i \right)^{\frac{1}{n}}$: This part of the formula looks somewhat complicated, but it is actually quite simple. It says:

1) precision_i : Find the individual precision values (n-grams that appear both in the hypothesis and in the reference) for 1-grams, 2-grams, 3-grams and 4-grams (the upper limit of 4 is not given in the formula, but the common BLEU score implementations used in MT research compute up to the n-gram value of 4). This will give you values such as $\frac{4}{7}$ (4 out of 7 possible 1-gram matches), $\frac{3}{6}$ (3 out of 6 possible 2-gram matches), $\frac{2}{5}$ (2 out of 5 possible 3-gram matches) and $\frac{1}{4}$ (1 out of 4 possible 4-gram matches).

2) $\left(\prod_{i=1}^n \text{precision}_i \right)^{\frac{1}{n}}$: Calculate the geometric mean of the n-gram precision values (F-Measure calculates the *harmonic* mean, BLEU calculates the *geometric* mean). To calculate the geometric mean, we multiply our n precision values and then take the n -th root of the product of these values. In our example, we have 1- to 4-grams (four precision values, therefore $n = 4$), so we multiply $\frac{4}{7} \times \frac{3}{6} \times \frac{2}{5} \times \frac{1}{4}$ and then take the fourth root of this product. Calculating the n -th root of a number is equivalent to raising this number to the exponent $\frac{1}{n}$ ($\frac{1}{4}$ in our example). So, our final calculation in this case would be $\left(\frac{4}{7} \times \frac{3}{6} \times \frac{2}{5} \times \frac{1}{4} \right)^{\frac{1}{4}}$.

Jupyter Notebooks

Formatted notebook document

2.1.2 BLEU

BLEU is an acronym for **Bi**Lingual **E**valuation **U**nderstudy. It was originally proposed in [Papineni et al. \(2002\): BLEU: A Method for Automatic Evaluation of Machine Translation](#). Today, it is still the most commonly used automatic MT quality score and it achieves reasonably good correlation with human quality judgements. BLEU is calculated using the following formula:

$$BLEU = \min(1, \frac{\text{hypothesis length}}{\text{reference length}}) (\prod_{i=1}^n \text{precision}_i)^{\frac{1}{n}}$$

Let's break this down:

$(\prod_{i=1}^n \text{precision}_i)^{\frac{1}{n}}$: This part of the formula looks somewhat complicated, but it is actually quite simple. It says:

1) precision_i : Find the individual precision values (n-grams that appear both in the hypothesis and in the reference) for 1-grams, 2-grams, 3-grams and 4-grams (the upper limit of 4 is not given in the formula, but the common BLEU score implementations used in MT research compute up to the n-gram value of 4). This will give you values such as $\frac{4}{7}$ (4 out of 7 possible 1-gram matches), $\frac{3}{6}$ (3 out of 6 possible 2-gram matches), $\frac{2}{5}$ (2 out of 5 possible 3-gram matches) and $\frac{1}{4}$ (1 out of 4 possible 4-gram matches).

2) $\prod_{i=1}^n$ and $\frac{1}{n}$: Calculate the geometric mean of the n-gram precision values (F-Measure calculates the *harmonic* mean, BLEU calculates the *geometric* mean). To calculate the geometric mean, we multiply our n precision values and then take the n -th root of the product of these values. In our example, we have 1- to 4-grams (four precision values, therefore $n = 4$), so we multiply $\frac{4}{7} \times \frac{3}{6} \times \frac{2}{5} \times \frac{1}{4}$ and then take the fourth root of this product. Calculating the n -th root of a number is equivalent to raising this number to the exponent $\frac{1}{n}$ ($\frac{1}{4}$ in our example). So, our final calculation in this case would be $(\frac{4}{7} \times \frac{3}{6} \times \frac{2}{5} \times \frac{1}{4})^{\frac{1}{4}}$.

Jupyter Notebooks

Example of teaching abstract MT concepts through Jupyter notebooks

2 Exploring individual word vectors

We can query our word embedding model/embedding matrix like a look-up table, i.e., we can ask the model to look up a certain word and it will return the word vector for this word. This is what the code below does. The output will be an array of 100 floating point numbers, each representing one vector dimension (remember, the vectors in this model are 100-dimensional). Let's look up the word vector for 'government', which is listed as one of the 150 most frequent words in the model. Feel free to modify this code to explore vectors for other words in our model.

```
# Display individual word vectors  
word_embeddings['government']
```

```
array([-0.219   , -0.15314 ,  0.028578,  0.19314 ,  0.017295,  0.2656  ,  
       -1.2929  ,  0.036583,  0.3522  ,  0.52552 , -0.013523,  0.47173 ,  
        0.31009 ,  0.29828 , -0.23724 ,  0.047662,  0.26025 , -0.44439 ,  
       -0.2429 , -0.079745,  0.39269 , -0.16625 ,  0.67831 , -0.3608  ,  
       -0.90875 , -0.59866 , -0.098991, -0.4032  , -0.25513 , -0.32609 ,  
        0.58892 ,  0.85139 , -0.31868 , -0.036752, -0.37764 ,  0.47531 ,  
        0.15249 ,  0.035057, -0.67195 , -0.24823 , -1.0172  ,  0.085827,  
        0.5829  ,  0.11308 ,  0.29078 , -0.49196 ,  0.080374, -0.44738 ,  
       -0.22613 , -0.58053 , -0.023165, -0.52726 ,  0.65555 ,  1.5924  ,  
       -0.34971 , -2.0056  , -0.063539, -0.14119 ,  2.587   ,  0.6543  ,  
        0.15131 , -0.69478 , -0.54288 ,  0.23742 ,  0.90566 ,  0.46563 ,  
       -0.33104 ,  0.62263 ,  0.56909 ,  0.12134 ,  0.43332 , -0.66784 ,  
       -0.69915 , -0.97821 ,  0.010897,  0.093153, -0.53922 ,  0.92222 ,  
       -1.8925  ,  0.3983  ,  1.1314  ,  0.15663 , -0.34177 , -0.099782,  
       -1.0973  , -0.026727,  0.28341 ,  0.6354  , -0.17618 , -1.0249  ,  
       -0.70725 , -0.74798 , -1.042   ,  0.17089 , -0.97143 ,  0.20341 ,  
        0.2177  , -0.40402 ,  0.45702 , -0.11144 ], dtype=float32)
```

Vector representations such as these are what a neural machine translation system actually processes when translating a sentence. You can think of our word embedding model/embedding matrix as a dictionary. We present the NMT system with a word such as 'government'. Since the system cannot process such natural language words, it will look up the vector of the word 'government' in the embedding matrix and use this vector representation for further processing.

The standard transformer architecture for neural machine translation actually works with a vector dimensionality of 512, so the vector representations in these systems will be about five times bigger than the vectors in our pretrained model.

(<https://colab.research.google.com/drive/1UUteTlvULD8mSh94Hd6pmL5OudP8l9bc?usp=sharing>)

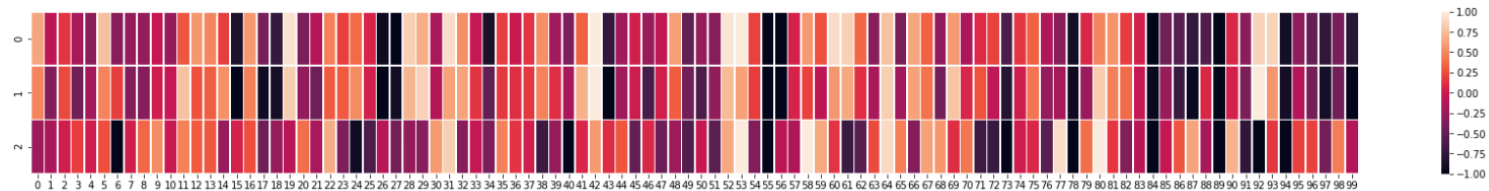
Jupyter Notebooks

Example of teaching abstract MT concepts through Jupyter notebooks – visualisations

If you run the code below, you will see a visual representation of the 100 dimensions of the similar words *father* and *son* together with the representation of the dissimilar word *government*. Create the representations and inspect the individual vector dimensions.

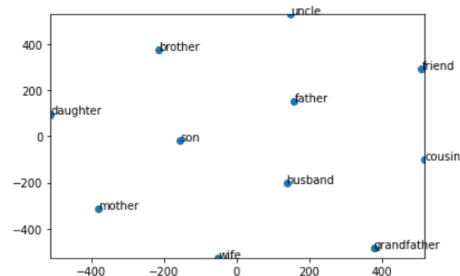
```
# Import the Libraries required to visualise the word embedding vectors
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Create the visual representations
plt.figure(figsize=(30,3))
sns.heatmap([word_embeddings["father"], word_embeddings["son"], word_embeddings["government"]], xticklabels=True, yticklabels=True, cbar=True,
            vmin=-1, vmax=1, linewidths=0.7)
plt.show()
```



Now, we are ready to visualize semantic relations between the words in our model. The following code visualizes the 10 words with exhibited the closest proximity to the word *father* in 100-dimensional vector space. Note that these are the same words that were listed when we queried our model for the 10 words most similar to *father* above.

```
[5] # Display word embeddings in two-dimensional space
display_tsne_reduction(word_embeddings, 'father')
```



Jupyter Notebooks

Setting up the notebook environment

0 Housekeeping

First, we need to ensure that we have the Natural Language Toolkit in its current version installed. Run the following code to test this.

```
# Upgrade to the current version of pip (if necessary) and check if NLTK is installed
!pip install --upgrade pip
!pip show nltk
```

If you get the message that the NLTK package was not found or an NLTK version older than 3.5 is displayed, run the code below. Otherwise, you can skip this step.

```
# Install NLTK or upgrade to its current version and run the nltk downloader
!pip install --upgrade nltk
import nltk
nltk.download()
```

If the NLTK Downloader prompts you to enter a command, enter 'd' and confirm with 'Shift + Enter'. In the 'Identifier' field, enter 'all-nltk' and confirm with 'Shift + Enter'. This downloads all available NLTK packages in their current versions. Once the download is finished, enter 'q' and confirm with 'Shift + Enter'.

NOTE: If a graphical downloader opens, just select 'all-nltk' from the 'Collections' list and click 'Download'.

We also need to install the Pyter package, which is not part of the Natural Language Toolkit but which we need for calculating the Translation Edit Rate. Just run the code below to install this package.

```
# Install the current version of the pyter package
!pip install --upgrade pyter3
```

Now you're set up and ready to go!

(<https://colab.research.google.com/drive/19Mub2IJV6SoyuvvsyjSeaGiSQ1q433jD?usp=sharing>)

GitHub Repository with Notebooks for MT Teaching

ITMK / MT_Teaching Unwatch 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

ITMK Add files via upload 185bfa8 on 12 Apr 35 commits

LICENSE	Initial commit	3 months ago
MT Quality Score Calculator_embeddi...	Add files via upload	2 months ago
MT Quality Score Calculator_tradition...	Add files via upload	3 months ago
README.md	Update README.md	2 months ago
Word Embeddings for NMT_fundame...	Add files via upload	2 months ago
Word Embeddings for NMT_large mo...	Add files via upload	2 months ago
requirements.txt	Update requirements.txt	2 months ago

README.md

Jupyter notebooks for machine translation technology teaching

This is a collection of Jupyter notebooks developed as teaching aids for machine translation teaching in the [MA in Specialised Translation](#) programme at the Institute of Translation and Multilingual Communication at TH Köln, Germany. The notebooks are provided as open-source resources under the MIT License and can be used by interested parties for their own translation technology classes.

About

A collection of Jupyter notebooks for teaching various aspects of machine translation technology to translation students.

ReadmeMIT License

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages

Jupyter Notebook 100.0%

(https://github.com/ITMK/MT_Teaching)

Pilot Study

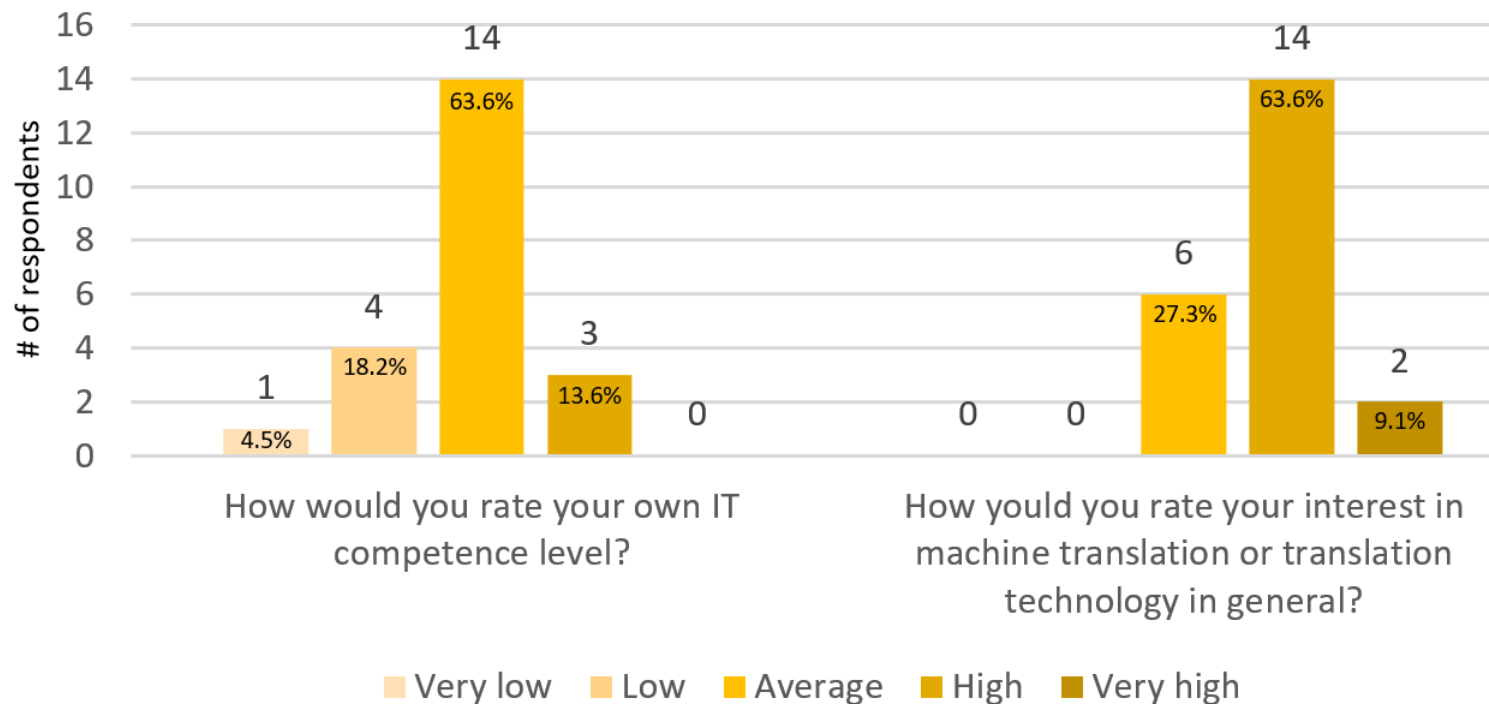
on the Use of Jupyter Notebooks in MT Teaching

- [MA in Specialised Translation](#) programme at the Institute of Translation and Multilingual Communication at TH Köln
- Course “Processes of Machine Translation”
- 25 students, 22 of which participated in the pilot study
- First, guided introduction to Jupyter notebooks
- Students then independently worked through the notebook [Understanding Word Embeddings for Neural Machine Translation – Fundamentals](#) and completed an online questionnaire afterwards

Pilot Study on the Use of Jupyter Notebooks in MT Teaching

Own Assessment of IT Competence and Interest in Translation Technology/MT

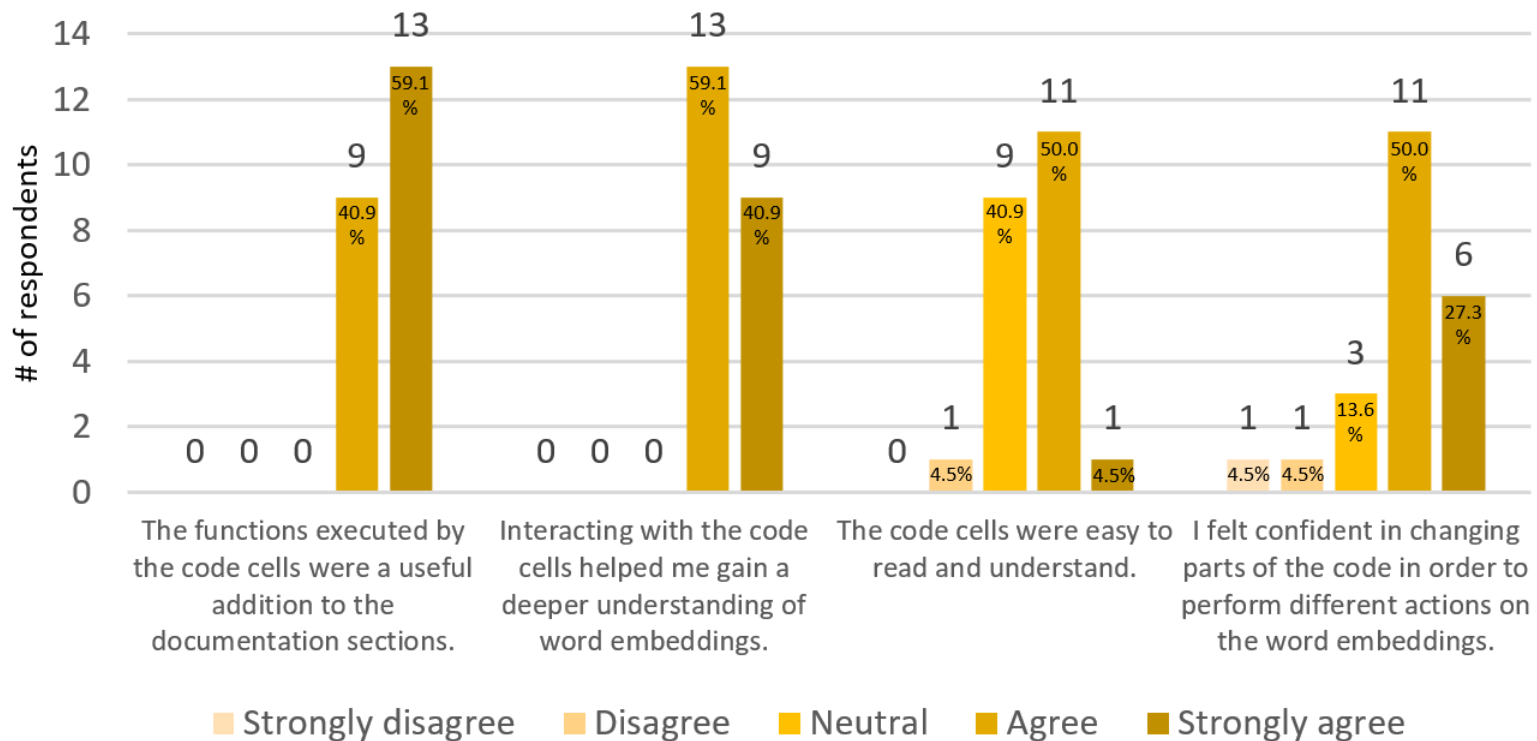
(absolute + percentage values, n=22)



Pilot Study on the Use of Jupyter Notebooks in MT Teaching

Learning Conceptual Knowledge through Jupyter Notebooks

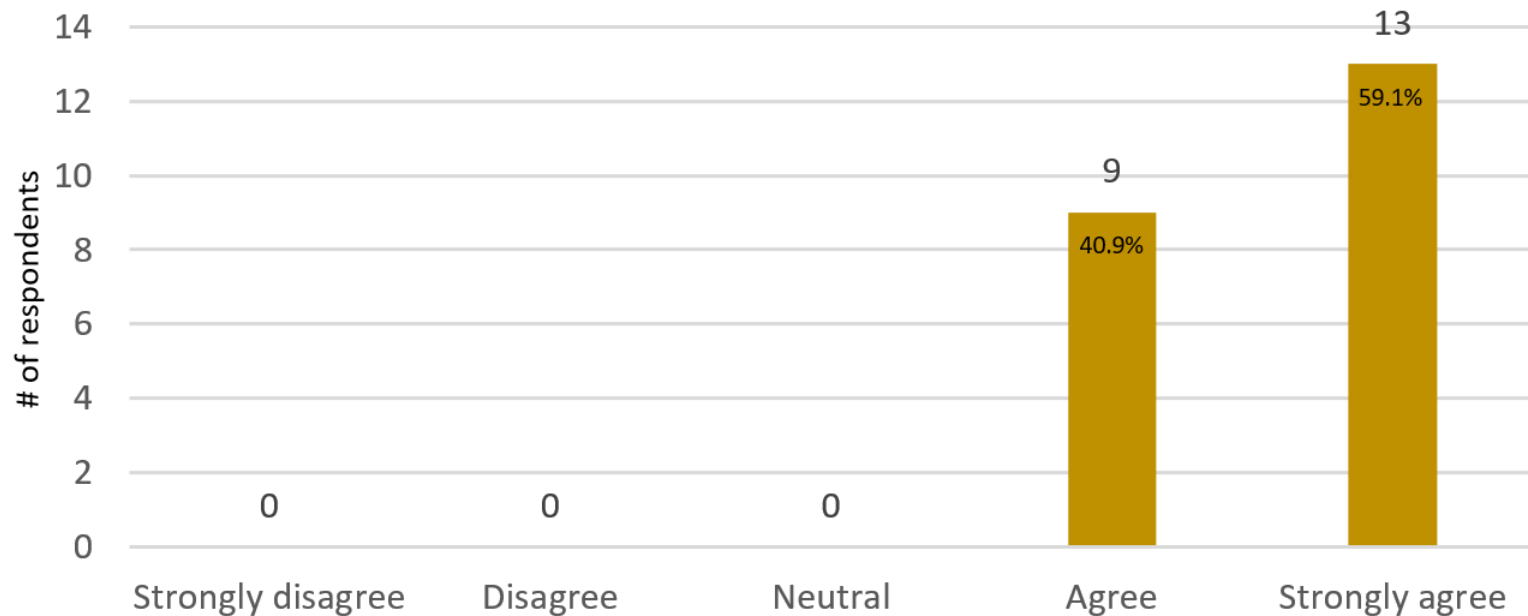
(absolute + percentage values, n=22)



Pilot Study on the Use of Jupyter Notebooks in MT Teaching

**Jupyter notebooks are a suitable additional resource
for teaching and learning translation technology/MT.**

(absolute + percentage values, n=22)



Pilot Study

on the Use of Jupyter Notebooks in MT Teaching

- examples of free comments:

P07: I had never worked with anything like Jupyter notebooks before but I found it quite intuitive to use. The possibility to interact with the code definitely helps with understanding and keeps me focused. It also makes abstract concepts easier to understand due to the 'visualisation' of those concepts.

P19: Students with a low IT competence level need the support of the teacher in order to use the notebook.

References

- Barba, Lorena A.; Barker, Lecia J.; Blank, Douglas S.; Brown, Jed; Downey, Allen B.; George, Timothy; Heagy, Lindsey J.; Mandli, Kyle T.; Moore, Jason K.; Lippert, David; Niemeyer, Kyle E.; Watkins, Ryan R.; West, Richard H.; Wickes, Elizabeth; Willing, Carol; Zingale, Michael (2019): *Teaching and Learning with Jupyter*.
<https://jupyter4edu.github.io/jupyter-edu-book/>
- EMT (2017): “European Master’s in Translation Competence Framework”, Website of the DG Translation of the European Commission. https://ec.europa.eu/info/sites/default/files/emt_competence_fwk_2017_en_web.pdf
- Jupyter Team (2015): “Jupyter Notebook User Documentation”. <https://jupyter-notebook.readthedocs.io/en/stable/index.html>.
- Millman, K. Jarrod; Pérez, Fernando (2014): “Developing Open Source Scientific Practice”, in: Stodden, Victoria; Leisch, Friedrich; Peng, Roger D. (eds): *Implementing Reproducible Research*. Boca Raton: Chapman and Hall/CRC, 149-183.
https://berkeley-stat159-f17.github.io/stat159-f17/_static/ref/millman-perez.pdf
- O’Brien, Sharon; Ehrensberger-Dow, Maureen (2020): “MT Literacy—A cognitive view”, in: *Translation, Cognition & Behaviour* 3(2), 145-164.