Tracing and visualizing diachronic semantic change using contextualized embeddings

Software project, group 5

Averie (Ho Zoen) SO, NGO Van Duy, Scott TANKARD, Mathilde AGUIAR

Université de Lorraine

Wednesday 1 Feb 2023

Outline

Introduction

Accomplishments

- Tuned BERT
- Tuned mBERT with EN and DE
- Extracted contextualised semantic meanings with both models
- Measured semantic shift of words across time periods
- Created a prototype of our system
- Launched the alpha version
- *Submitted the report*

Shift measurements, eval scores

SemEval Results

model	**emb. type**	**English**
Schlechtweg et al (2019) [1]	SGNS	0.321
Pomsl and Lyapin (2020) [2]	SGNS	0.422
Montariol et al (2021) [3]	bert	0.456
our model (En only)	mbert	0.408
our model (En + De)	mbert	0.421

	k-means 5	**k-means 7**	**AP**
Montariol et al (2021)	0.375	-	0.437
our model (En only)	0.408	0.384	0.354
our model (En + De)	0.421	0.383	0.340

Table: Spearman correlation with WD as measurement.

[1] previous SOTA employing non-contextual word embeddings
[2] Winner of SemEval 2020 Task 1, subtask 2
[3] scalable_semantic_shift

SemEval results - observations

- Compared across our two models, **k-means 5** consistently and significantly outperforms the two other methods - k-means 7 and Affinity Propagation (AP), in this order, while the original implementation did not report a single best clustering method across different models.

Web app frontend

Web app frontend

- Removed the tree visualization $\rightarrow$ added a table visualization instead
- User query implemented
- Various updates on the interfaces

`http://127.0.0.1:5000/`

Pipeline backend

- The conceptual steps of the pipeline
  - Corpus Preparation
  - Fine-tuning BERT Model
  - Embeddings Extraction
  - Measuring Semantic Shift
    (In the final step, a dual output of evaluation metrics for automatic evaluation against gold-standard data, and data files to be fed into the visualisation component.)
  - Afterwards: Evaluation and/or visualisation
- The pipeline backend components
  - reduce_corpus.py
  - measure_semantic_shift_merged.py
  - measure_semantic_shift_visualisation.py
  - g5_tools.py
  - And scripts from the original SSS (scalable_semantic_shift) codebase:
    - `build_coha_corpus.py`
    - `fine-tune_BERT.py`
    - `get_embeddings_scalable.py`

reduce_corpus.py

- Filter/reduce the dataset (according to occurrences of query target words of interest)
- Downsample the dataset (sample a subset) based on desired corpus size limits
- Why: running an embeddings-extraction operation can take quite a while (often 1-5 hours; in some cases, as long as 44 hours), depending on size of dataset and length of target wordlist
- With filtering+downsampling, we are able to bring down the time required for an extract from many hours to just minutes (often 20 minutes for a small wordlist)

measure_semantic_shift_merged.py and measure_semantic_shift_visualisation.py

- `measure_semantic_shift_merged.py`:
  - Code for running measurement metrics on the extracted embeddings, merging the pickle files from different time slices
  - Clustering and measuring shift in a probability distribution. Metrics: JSD (Jensen-Shannon divergence) and WD (Wasserstein distance).
  - Train KMeans models over the embeddings; combine all the slice-based clusters into one; compute differences between the slices.
  - Results exported to CSV file
- `measure_semantic_shift_visualisation.py`:
  - Code for producing semantic distance measurements useful for the visualisation component: Comparing distance from neighbours across time
  - Retrieve embeddings for most frequent sense of word; compute the distances from that word to the rest of the vocabulary.
  - Sort the distances; obtain the `n` closest neighbours of the word
  - Results exported to CSV for visualisation on web app.

g5_tools.py

- Bringing it all together: A set of wrappers, helper functions and command-line tools, tying together the various components of the pipeline
- Calls out to our own scripts as well as to the original SSS scripts
- We also implement:
  - extensive and detailed logging (useful for reproducibility of experiments, unattended runs, troubleshooting, and measuring time and computing resources spent),
  - capture of subcommand output (stderr/stdout),
  - checksumming of inputs and outputs,
  - ...and other mechanisms, in a flexible and modular framework.

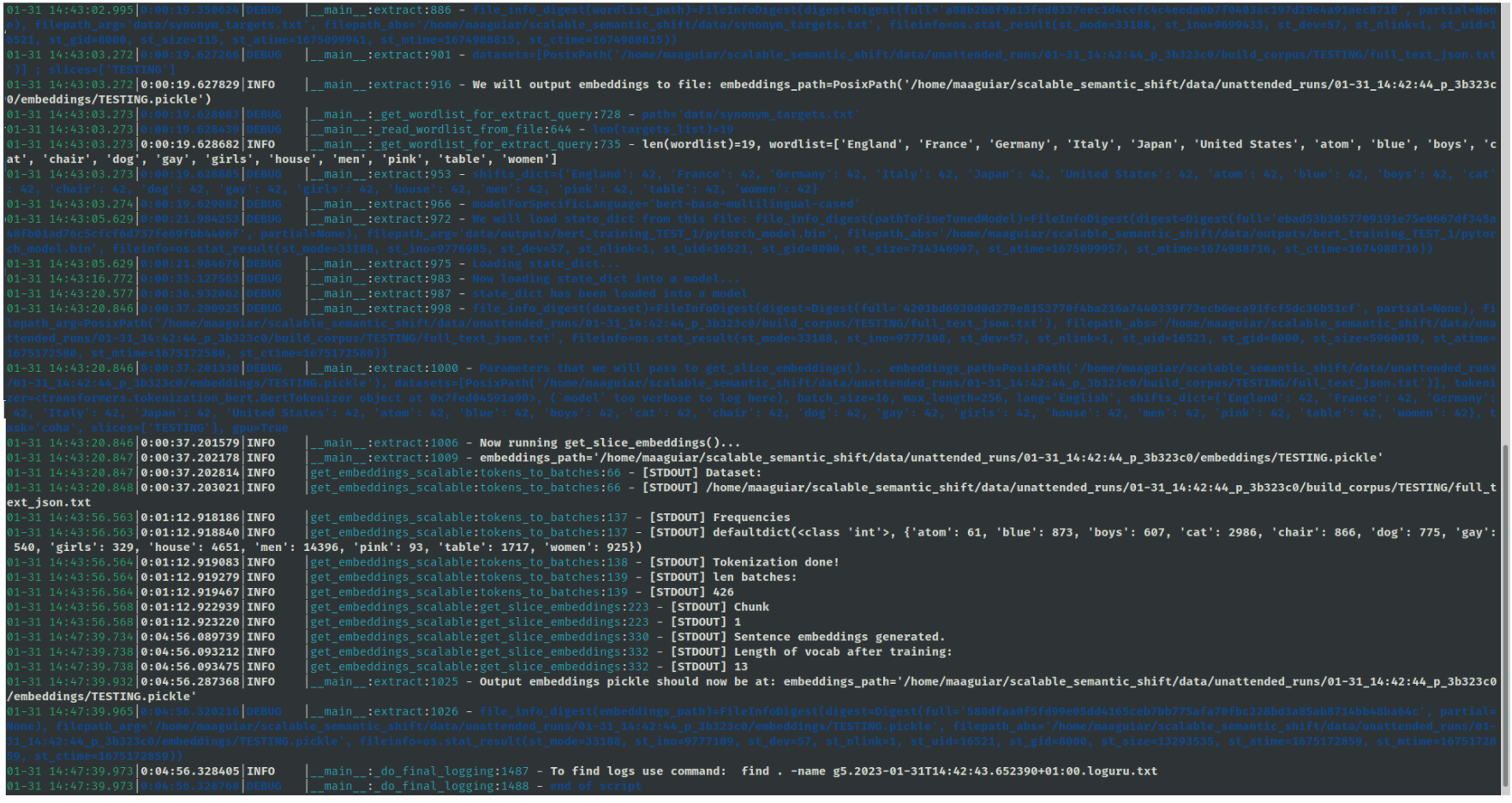

Figure: Screenshot of running filter_dataset_and_extract operation using g5_tools.py

Conclusion

Conclusions

- Coming next: Session 8: 7 Feb. 2023 9h00 to 17h00 Final project presentations
- We have set up two languages, for two time periods. Future work:
  - Future: extend this work to more than two time periods, since our visualisation tool is advantageous in presenting complex patterns that are not easily interpreted by numbers.
  - Add additional languages into the corpuses and re-train the models.
  - Add cleaning or standardisation procedures for orthographic differences since a change in spelling in the same words can be a common phenomenon, especially for texts from a long time ago, and in certain languages such as English.
  - Running experiments comparing multiple-senses *vs* single-averaged sense;
  - testing on different types of semantic change;
  - analyzing multiple languages in comparison to each other (e.g. evolution of Sir/Monsieur in English/French);
  - deepening and enriching the historical event contextualization data that we currently present in the UI;
  - future semantic change prediction.

Thank you!

Question time