Digital Humanities & Research Software Engineering working together

Some examples of a fruitful collaboration from the Living with Machines project

Kaspar von Beelen, Mariona Coll Ardanuy, Kasra Hosseini and Federico Nanni
The Alan Turing Institute
Overview of the Talk

1. The Research Engineering Group
2. The Living with Machines Project
3. How we work together (in theory)
4. How we work together (in practice)
   a. The “Atypical Animacy” Project
   b. DeezyMatch
5. Lessons learned
The Research Engineering Group
Turing REG

- A team of ~35 research software engineers and research data scientists
- Range of backgrounds: physics, biology, computer science, psychology, mathematics, digital humanities...
- Enthusiastic collaborators, researchers and developers who want to make long-lasting, reproducible and robust tools and analyses
Turing Challenges

- Revolutionise healthcare
- Deliver safer, smarter engineering
- Manage security in an insecure world
- Shine a light on our economy
- Make algorithmic systems fair, transparent, and ethical
- Design computers for the next generation of algorithms
- Supercharge research in science and humanities
- Foster government innovation
Projects

- Projects can last anywhere from a few months to >1 year.
- Projects can come from Turing Fellows, industry partners or be generated internally - recent partners include: [logos]

- Range from purely “data science” to purely “software development”, or anywhere in-between.
Living with Machines
Living with Machines

- Funded by the AHRC as part of the UKRI Strategic Priorities Fund
- Collaboration between the Alan Turing Institute and the British Library
- Partner institutions: Cambridge, East Anglia, Exeter, Queen Mary
Massive Interdisciplinary Collaboration

Professor Ruth Ahnert
Turing Fellow

David Beavan
Senior Research Software Engineer – Digital Humanities

Professor Emma Griffin
Turing Fellow

Professor Jon Lawrence
University of Exeter

Dr Katherine McDonough
Senior Research Associate

Dr Federico Nanni
Research Data Scientist

André Pliza
Research Project Manager, Data Science for Science

Karen Cordier
Research Project Manager (Parental Leave Cover), Living with Machines

Dr Barbara McGillivray
Turing Research Fellow

Maja Maricevic
British Library

Dr Mia Ridge
British Library

Sir Alan Wilson
Director, Special Projects

Dr Giorgia Tolfo
Data and Content Manager, British Library

Dr Olivia Vane
Researcher, British Library

Daniel van Strien
Digital Curator, British Library

Dr Daniel Wilson
Research Associate

Dr Giovanni Colavizza
Visiting Researcher

Dr Adam Farquhar
British Library

Dr James Hetherington
Director of Data Science in Practice

Dr Yann Ryan
British Library

Dr Joshua Rhodes
Research Associate

Dr Sarah Gibson
Research Software Engineer

Dr Rosa Filgueira
Data Architect, EPCC

Dr Timothy Hobson
Senior Research Software Engineer
Living with Machines is...

- An inquiry into how technology impacted the lives of “ordinary people” in Britain 1780-1914 (history from below)
- A study of the ever-changing relation between humans and technology
- Explores the social and cultural impact of the Industrial Revolution by mining (massive) historical collections (newspaper, maps, census)
Living with Machines

- Applies computational methods to a domain (history) that has an uncomfortable relation with quantification

- An investigation into what it means to use computational analysis for history
Living with Machines

- Explore heritage collections at **scale**:
  - “Distant” vs “close” reading
- **Linking** historical sources
Living with Machines

Radical collaboration:

- Power imbalances related to knowledge and expertise
- Different intellectual traditions and priorities
  - The problem of putting (binary) labels on things
- Different levels of technical skills and domain expertise within the team (“scattered expertise”)
How we work together (in theory!)
Example timeline

Week(s)

Start
- Hypothesis, initial questions
- Turn ideas into DS / SE (sub-)tasks
- EDA: Exploratory Data Analysis
- Design flowchart

Hack week(s)
- Exploring available datasets / libraries / methods
- Identify limitations of existing methods or datasets
- Do we need new method(s) for our tasks?
- Collect more data?
- Need annotations?
- Other requirements

Month(s)

Set up infrastructure
- Repositories (e.g., GitHub), project board for planning / sprint meetings
- Public repos from the beginning (if at all possible)
- Create DB
- Baselines
- Start with simple or well-established methods
- Start writing?

Implement new methods
- Test, CI, reproducibility, develop a new library?

Results and analysis
- Write paper

Submit paper
- Make repo public? (documentation, env, ...)

Maintenance
- Other users, projects?
- Address issues, add more functionalities, ...
Example timeline

Week(s)

Start
Hypothesis, initial questions

Turn ideas into DS / SE (sub-)tasks

EDA Exploratory Data Analysis

Design flowchart

Hack week(s)
Exploring available datasets / libraries / methods

Identify limitations of existing methods or datasets

Do we need new method(s) for our tasks?

Collect more data?

Need annotations?

Other requirements

Month(s)

Set up infrastructure
Repositories (e.g., GitHub), project board for planning / sprint meetings

Public repos from the beginning (if at all possible)

Create DB

Baselines
Start with simple or well-established methods

Start writing?

Deadline is approaching!!!

Implement new methods
Test, CI, reproducibility, develop a new library?

Results and analysis
Write paper

Submit paper
Make repo public? (documentation, env, …)

Maintenance
Other users, projects?

Address issues, add more functionalities, …

year(s)
How we work together (in practice!)
Living Machines
A Study of Atypical Animacy
Animacy in Linguistics

- Animacy is the **property of being alive**
- **Linguistic animacy** of a given entity tends to align with its biological animacy

... *but not always:*

“*He exclaimed; the machine has heard you: it moves!*”

*The Penny Library of Famous Books, 1895, Publ: George Newnes*

- **Machines** sit at the fuzzy **boundary** between animacy and inanimacy (Yamamoto, 1999): deliberate or unconscious
19thC Britain: a society being transformed by industrialization

How machines have been imagined in the 19th century from lifeless mechanical objects to living beings, and even human-like agents that feel, think, kill, and love

Trace this phenomenon at scale: through time, space, ideologies

Relevant for today’s discussion of the impact of technology in our society (Alan Turing, 1950: “Can machines think?”)
19thC Machines animacy dataset

Gathering data to annotate

- Goal: create a dataset of animacy of machines
- Original corpus: 19thC BL Books, ≈48,200, ≈4.9B tokens
- We extracted sentences in English containing machine words ("machine, engine, locomotive...")
- We extracted interesting sentences through pooling using different methods
Animacy (true/false): true if the machine is represented as having traits or characteristics (maybe implicit) distinctive of biologically animate beings or human-specific skills, feelings, or emotion.

Humanness (true/false): true if the machine is represented as sentient and capable of specifically human emotions.

“No, no, to her mother poor Fraulein was not a woman, a heart, a soul; she was just a machine.”

Into an Unknown World. A novel, 1897, J.S. Winter
593 sentences: 201 animate/292 inanimate expressions
Krippendorff’s $\alpha=0.74$ on animacy, $\alpha=0.50$ on humanness.
Rich in atypical animacy.

<table>
<thead>
<tr>
<th>Target</th>
<th>Sentence</th>
<th>Animacy</th>
<th>Humanness</th>
</tr>
</thead>
<tbody>
<tr>
<td>engine</td>
<td>In December, the first steam fire <strong>engine</strong> was received, and tried on the shore of Lake Monona, with one thousand feet of hose.</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>engine</td>
<td>It was not necessary for Jakic to slow down in order to allow the wild <strong>engine</strong> to come up with him; she was coming up at every revolution of her wheels.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>locomotive</td>
<td>Nearly a generation had been strangely neglected to grow up un-Americanized, and the private adventurer and the <strong>locomotive</strong> were the untechnical missionaries to open a way for the common school.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>machine</td>
<td>The worst of it was, the people were surly; not one would get out of our way until the last minute, and many pretended not to see us coming, though the <strong>machine</strong>, held in by the brake, squeaked a pitiful warning.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>machines</td>
<td>Our servants, like mere <strong>machines</strong>, move on their mercenary track without feeling.</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>machinery</td>
<td>We have everywhere water power to any desirable extent, suitable for propelling all kinds of <strong>machinery</strong>.</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Approach in a nutshell

Joy and sorrow - life and death, wrote the little machine.

Joy and sorrow - life and death, wrote the little [MASK].

BERT, predict the missing word in the sentence:
Determining animacy

- **Assumption:** given a context requiring an animate entity, a contextualized LM predicts tokens corresponding to _conventionally_ animate entities.

- For each token in top predicted tokens:
  - Disambiguate to most probable WordNet sense
  - Determine the animacy of the sense using Wordnet hierarchy of nouns

- Threshold and cutoff are found through experimentation.
A Language Model is meant to be a faithful representation of the language that has been used to train it.

“They were told that the [MASK] stopped working.”

**BERT language models trained on...**

<table>
<thead>
<tr>
<th>Pre 1850 text:</th>
<th>1850-1875 text:</th>
<th>1875-1890 text:</th>
<th>1890-1900 text:</th>
</tr>
</thead>
<tbody>
<tr>
<td>man 5.3291</td>
<td>men 10.7655</td>
<td>men 10.2048</td>
<td>mercury 8.0446</td>
</tr>
<tr>
<td>prisoners 4.9758</td>
<td>people 9.497</td>
<td>miners 7.6654</td>
<td>machinery 7.4067</td>
</tr>
<tr>
<td>men 4.885</td>
<td>miners 9.249</td>
<td>machines 7.4062</td>
<td>machine 7.2903</td>
</tr>
<tr>
<td>book 4.6477</td>
<td>engine 8.0428</td>
<td>people 7.2991</td>
<td>mine 7.274</td>
</tr>
<tr>
<td>people 4.556</td>
<td>women 8.0126</td>
<td>engine 7.232</td>
<td>mill 7.057</td>
</tr>
<tr>
<td>one 4.4271</td>
<td>company 7.7261</td>
<td>labourers 7.0957</td>
<td>men 7.0257</td>
</tr>
<tr>
<td>slaves 4.4034</td>
<td>machine 7.6021</td>
<td>engines 6.7786</td>
<td>engine 6.9966</td>
</tr>
<tr>
<td>air 4.1329</td>
<td>labourers 7.5987</td>
<td>engineers 6.5642</td>
<td>lead 6.9177</td>
</tr>
<tr>
<td>water 4.1148</td>
<td>machines 7.5012</td>
<td>machine 6.4712</td>
<td>miners 6.7764</td>
</tr>
</tbody>
</table>
Experiments: baselines

- Most frequent class
- Classification approach
  - Classifiers: SVMs (word embeddings, TFIDF) and BERT Classifier
  - Inputs:
    - \( \text{targetExp} \): target expression
    - \( \text{targetExp} + \text{ctxt} \): target expression + context (3 token left and right)
    - \( \text{maskedExp} + \text{ctxt} \): masked target expression + context (3 token left and right)
- LSTM sequential tagging approach
## Results

<table>
<thead>
<tr>
<th>Most frequent class</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM TFIDF: targetExp</td>
<td>0.911</td>
<td>0.893</td>
<td>0.902</td>
<td>0.928</td>
<td>0.696</td>
<td>0.713</td>
<td>0.704</td>
<td>0.474</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp</td>
<td>0.927</td>
<td>0.919</td>
<td>0.923</td>
<td>0.954</td>
<td>0.694</td>
<td>0.711</td>
<td>0.702</td>
<td>0.499</td>
</tr>
<tr>
<td>BERTClassifier: targetExp</td>
<td>0.951</td>
<td><strong>0.948</strong></td>
<td>0.949</td>
<td><strong>0.985</strong></td>
<td>0.698</td>
<td>0.715</td>
<td>0.706</td>
<td>0.51</td>
</tr>
<tr>
<td>SVM TFIDF: targetExp + ctxt</td>
<td>0.734</td>
<td>0.739</td>
<td>0.737</td>
<td>0.859</td>
<td>0.688</td>
<td>0.71</td>
<td>0.699</td>
<td>0.651</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp + ctxt</td>
<td>0.758</td>
<td>0.742</td>
<td>0.75</td>
<td>0.876</td>
<td>0.728</td>
<td>0.531</td>
<td>0.614</td>
<td>0.481</td>
</tr>
<tr>
<td>BERTClassifier: targetExp + ctxt</td>
<td>0.931</td>
<td>0.926</td>
<td>0.929</td>
<td>0.978</td>
<td>0.695</td>
<td>0.721</td>
<td>0.708</td>
<td>0.721</td>
</tr>
<tr>
<td>SVM TFIDF: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.677</td>
<td>0.675</td>
<td>0.804</td>
<td>0.592</td>
<td>0.6</td>
<td>0.596</td>
<td>0.498</td>
</tr>
<tr>
<td>SVM WordEmb: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.678</td>
<td>0.676</td>
<td>0.809</td>
<td>0.518</td>
<td>0.52</td>
<td>0.519</td>
<td>0.339</td>
</tr>
<tr>
<td>BERTClassifier: maskedExp + ctxt</td>
<td>0.855</td>
<td>0.852</td>
<td>0.854</td>
<td>0.951</td>
<td>0.687</td>
<td>0.696</td>
<td>0.692</td>
<td>0.603</td>
</tr>
<tr>
<td>SeqModel: LSTM</td>
<td><strong>0.952</strong></td>
<td><strong>0.948</strong></td>
<td><strong>0.95</strong></td>
<td>0.949</td>
<td>0.697</td>
<td>0.719</td>
<td>0.708</td>
<td>0.482</td>
</tr>
<tr>
<td>MaskPredict: BERT-base</td>
<td>0.739</td>
<td>0.703</td>
<td>0.72</td>
<td>0.848</td>
<td>0.719</td>
<td>0.742</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td>MaskPredict: BERT-base +ctxt</td>
<td>0.839</td>
<td>0.774</td>
<td>0.806</td>
<td>0.892</td>
<td>0.758</td>
<td><strong>0.778</strong></td>
<td>0.768</td>
<td><strong>0.795</strong></td>
</tr>
<tr>
<td>MaskPredict: fit19thBERT +ctxt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.758</td>
<td>0.775</td>
<td>0.766</td>
<td>0.777</td>
</tr>
<tr>
<td>MaskPredict: early19thBERT +ctxt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.799</strong></td>
<td>0.773</td>
<td><strong>0.786</strong></td>
<td>0.784</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Most frequent class</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM TFIDF: targetExp</td>
<td>0.911</td>
<td>0.893</td>
<td>0.902</td>
<td>0.928</td>
<td>0.696</td>
<td>0.713</td>
<td>0.704</td>
<td>0.474</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp</td>
<td>0.927</td>
<td>0.919</td>
<td>0.923</td>
<td>0.954</td>
<td>0.694</td>
<td>0.711</td>
<td>0.702</td>
<td>0.499</td>
</tr>
<tr>
<td>BERTClassifier: targetExp</td>
<td>0.951</td>
<td><strong>0.948</strong></td>
<td>0.949</td>
<td><strong>0.985</strong></td>
<td>0.698</td>
<td>0.715</td>
<td>0.706</td>
<td>0.51</td>
</tr>
<tr>
<td>SVM TFIDF: targetExp + ctxt</td>
<td>0.734</td>
<td>0.739</td>
<td>0.737</td>
<td>0.859</td>
<td>0.688</td>
<td>0.71</td>
<td>0.699</td>
<td>0.651</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp + ctxt</td>
<td>0.758</td>
<td>0.742</td>
<td>0.75</td>
<td>0.876</td>
<td>0.728</td>
<td>0.531</td>
<td>0.614</td>
<td>0.481</td>
</tr>
<tr>
<td>BERTClassifier: targetExp + ctxt</td>
<td>0.931</td>
<td>0.926</td>
<td>0.929</td>
<td>0.978</td>
<td>0.695</td>
<td>0.721</td>
<td>0.708</td>
<td>0.721</td>
</tr>
<tr>
<td>SVM TFIDF: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.677</td>
<td>0.675</td>
<td>0.804</td>
<td>0.592</td>
<td>0.6</td>
<td>0.596</td>
<td>0.498</td>
</tr>
<tr>
<td>SVM WordEmb: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.678</td>
<td>0.676</td>
<td>0.809</td>
<td>0.518</td>
<td>0.52</td>
<td>0.519</td>
<td>0.339</td>
</tr>
<tr>
<td>BERTClassifier: maskedExp + ctxt</td>
<td>0.855</td>
<td>0.852</td>
<td>0.854</td>
<td>0.951</td>
<td>0.687</td>
<td>0.696</td>
<td>0.692</td>
<td>0.603</td>
</tr>
<tr>
<td>SeqModel: LSTM</td>
<td><strong>0.952</strong></td>
<td><strong>0.948</strong></td>
<td><strong>0.95</strong></td>
<td><strong>0.949</strong></td>
<td>0.697</td>
<td>0.719</td>
<td>0.708</td>
<td>0.482</td>
</tr>
<tr>
<td>MaskPredict: BERT-base</td>
<td>0.739</td>
<td>0.703</td>
<td>0.72</td>
<td>0.848</td>
<td>0.719</td>
<td>0.742</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td>MaskPredict: BERT-base +ctxt</td>
<td>0.839</td>
<td>0.774</td>
<td>0.806</td>
<td>0.892</td>
<td>0.758</td>
<td><strong>0.778</strong></td>
<td>0.768</td>
<td><strong>0.795</strong></td>
</tr>
<tr>
<td>MaskPredict: fit19thBERT +ctxt</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>0.758</td>
<td>0.775</td>
<td>0.766</td>
<td>0.777</td>
</tr>
<tr>
<td>MaskPredict: early19thBERT +ctxt</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td><strong>0.799</strong></td>
<td>0.773</td>
<td><strong>0.786</strong></td>
<td>0.784</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Model Configuration</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Map</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Most frequent class</strong></td>
<td>0.31</td>
<td>0.5</td>
<td>0.383</td>
<td>0.623</td>
<td>0.336</td>
<td>0.5</td>
<td>0.402</td>
<td>0.318</td>
</tr>
<tr>
<td>SVM TFIDF: targetExp</td>
<td>0.911</td>
<td>0.893</td>
<td>0.902</td>
<td>0.928</td>
<td>0.696</td>
<td>0.713</td>
<td>0.704</td>
<td>0.474</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp</td>
<td>0.927</td>
<td>0.919</td>
<td>0.923</td>
<td>0.954</td>
<td>0.694</td>
<td>0.711</td>
<td>0.702</td>
<td>0.499</td>
</tr>
<tr>
<td>BERTClassifier: targetExp</td>
<td>0.951</td>
<td>0.948</td>
<td>0.949</td>
<td><strong>0.985</strong></td>
<td>0.698</td>
<td>0.715</td>
<td>0.706</td>
<td>0.51</td>
</tr>
<tr>
<td>SVM TFIDF: targetExp + ctxt</td>
<td>0.734</td>
<td>0.739</td>
<td>0.737</td>
<td>0.859</td>
<td>0.688</td>
<td>0.71</td>
<td>0.699</td>
<td>0.651</td>
</tr>
<tr>
<td>SVM WordEmb: targetExp + ctxt</td>
<td>0.758</td>
<td>0.742</td>
<td>0.75</td>
<td>0.876</td>
<td>0.728</td>
<td>0.531</td>
<td>0.614</td>
<td>0.481</td>
</tr>
<tr>
<td>BERTClassifier: targetExp + ctxt</td>
<td>0.931</td>
<td>0.926</td>
<td>0.929</td>
<td>0.978</td>
<td>0.695</td>
<td>0.721</td>
<td>0.708</td>
<td>0.721</td>
</tr>
<tr>
<td>SVM TFIDF: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.677</td>
<td>0.675</td>
<td>0.804</td>
<td>0.592</td>
<td>0.6</td>
<td>0.597</td>
<td>0.498</td>
</tr>
<tr>
<td>SVM WordEmb: maskedExp + ctxt</td>
<td>0.674</td>
<td>0.678</td>
<td>0.676</td>
<td>0.809</td>
<td>0.518</td>
<td>0.52</td>
<td>0.519</td>
<td>0.339</td>
</tr>
<tr>
<td>BERTClassifier: maskedExp + ctxt</td>
<td>0.855</td>
<td>0.852</td>
<td>0.854</td>
<td>0.951</td>
<td>0.687</td>
<td>0.696</td>
<td>0.692</td>
<td>0.603</td>
</tr>
<tr>
<td>SeqModel: LSTM</td>
<td><strong>0.952</strong></td>
<td><strong>0.948</strong></td>
<td><strong>0.95</strong></td>
<td>0.949</td>
<td>0.697</td>
<td>0.719</td>
<td>0.708</td>
<td>0.482</td>
</tr>
<tr>
<td>MaskPredict: BERT-base</td>
<td>0.739</td>
<td>0.703</td>
<td>0.72</td>
<td>0.848</td>
<td>0.719</td>
<td>0.742</td>
<td>0.73</td>
<td>0.74</td>
</tr>
<tr>
<td>MaskPredict: BERT-base +ctxt</td>
<td>0.839</td>
<td>0.774</td>
<td>0.806</td>
<td>0.892</td>
<td>0.758</td>
<td><strong>0.778</strong></td>
<td>0.768</td>
<td><strong>0.795</strong></td>
</tr>
<tr>
<td>MaskPredict: fit19thBERT +ctxt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.758</td>
<td>0.775</td>
<td>0.766</td>
<td>0.777</td>
</tr>
<tr>
<td>MaskPredict: early19thBERT +ctxt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td><strong>0.799</strong></td>
<td>0.773</td>
<td><strong>0.786</strong></td>
<td>0.784</td>
</tr>
</tbody>
</table>
A Reproducible Experimental Setting

Living Machines:
A Study of Atypical Animacy

This repository provides underlying code and materials for the paper 'Living Machines: A Study of Atypical Animacy' (COLING2020).

Table of contents

- Installation
- Directory structure
- Description of the codes
- Datasets and resources
- Evaluation results
- Citation
- Acknowledgements
- License

https://github.com/Living-with-machines/AtypicalAnimacy
Future work

- Develop new methods for targeted sense disambiguation for conducting animacy detection at scale
- Distinction between animacy and humanness
  - relation with the process of dehumanization through the language of mechanization
- Examine biases and social changes embedded in the language models
- In-depth study of the contextual cues that grant animacy and humanness.
DeezyMatch: A Deep Learning Approach to Fuzzy String Matching for Entity Linking
Motivation

Place names identified in news articles that refer to **Ashton-under-Lyne**:

Ashton-under-Lyne
Ashtonunder-line
ASHTONCLUDER-LYNE
Ashton-under-lyne
Ashtonunder-Lyne
ASHTON-UXDER-LYNE
Ashton-cnder-Lyne
Aditon-under-line
Asbtcn-under-Lyne
Ashton
ASHTON-UNDER-LYNE

Problem

We want to link to a knowledge base (e.g. Wikidata)
- But high degree of name variation!!
- And there are 822,161 Wikidata UK place names

<table>
<thead>
<tr>
<th>Ashton-under-Lyne (Q659803)</th>
</tr>
</thead>
<tbody>
<tr>
<td>market town in the Metropolitan Borough of Tameside, Greater Manchester, England</td>
</tr>
</tbody>
</table>

**Traditional** approaches to (fuzzy) string matching:

1. Exact string matching
2. Calculate string similarity between a query and the 822,161 potential place names, and sort by most similar candidates: very **time consuming**!!
DeezyMatch: introduction

A flexible deep learning approach to fuzzy string matching and candidate ranking.
DeezyMatch: architecture

A flexible deep learning approach to fuzzy string matching and candidate ranking.

Hosseini et al. (2020)
DeezyMatch: architecture

A flexible deep learning approach to fuzzy string matching and candidate ranking.
DeezyMatch: architecture

A flexible deep learning approach to fuzzy string matching and candidate ranking.

Vectors for "all candidate mentions" are computed **only once**

Adaptive searching **algorithm** applicable to large KBs and query sets

Hosseini et al. (2020)
DeezyMatch: features

A free, open-source software library written in Python for fuzzy string matching and candidate ranking:

- Easy-to-use interface
- Various deep neural network architectures for training new classifiers.
- User can change the architecture (RNN, GRU or LSTM), hyperparameters and preprocessing steps via input file.

```python
from DeezyMatch import train
from DeezyMatch import inference

# train a new model
train(input_file_path,
     dataset_train_path,
     model_name)

# model inference
inference(input_file_path,
          dataset_inference_path,
          pretrained_model_path)
```

Hosseini et al. (2020)
DeezyMatch: features

A free, open-source software library written in Python for fuzzy string matching and candidate ranking:
- Easy-to-use interface
- Various deep neural network architectures for training new classifiers.
- User can change the architecture (RNN, GRU or LSTM), hyperparameters and preprocessing steps via input file.

Hosseini et al. (2020)

Train: Dataset-1
Eval : Dataset-2

Train: Dataset-2
Eval : Dataset-2

Train: Dataset-1 + 2
Eval : Dataset-2

Train: Dataset-1
Eval : Dataset-2

F1 Score

#entries
DeezyMatch: features

A **free, open-source** software library written in Python for fuzzy string matching and candidate ranking:

- Easy-to-use interface
- Various deep neural network architectures for training new classifiers.
- User can change the architecture (RNN, GRU or LSTM), hyperparameters and preprocessing steps via input file.
- Fine-tuning a pretrained model; transfer learning.
- Extensive documentation: [https://github.com/Living-with-machines/DeezyMatch](https://github.com/Living-with-machines/DeezyMatch)

```python
from DeezyMatch import train
from DeezyMatch import inference

# train a new model
train(input_file_path, dataset_train_path, model_name)

# model inference
inference(input_file_path, dataset_inference_path, pretrained_model_path)
```

![Graph](image.png)

Hosseini et al. (2020)
DeezyMatch: performance

**Pair-classifier** performance as measured by F-score compared with other methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Santos</th>
<th>WG:en</th>
<th>OCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>LevDam</td>
<td>0.70</td>
<td>0.74</td>
<td>0.76</td>
</tr>
<tr>
<td>Santos et al. (2018a)</td>
<td>0.82</td>
<td>0.92</td>
<td>0.95</td>
</tr>
<tr>
<td>DeezyMatch</td>
<td>0.89</td>
<td>0.94</td>
<td>0.95</td>
</tr>
</tbody>
</table>

DeezyMatch (DM) **candidate ranker** performance compared to LevDam(LD) and exact. T/q: "Time per query" on CPU.

<table>
<thead>
<tr>
<th>Method</th>
<th>P@1</th>
<th>MAP@10</th>
<th>MAP@20</th>
<th>T/q</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgM:exact</td>
<td>0.69</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ArgM:LD</td>
<td>0.78</td>
<td>0.72</td>
<td>0.70</td>
<td>9s</td>
</tr>
<tr>
<td>ArgM:DM</td>
<td>0.78</td>
<td>0.76</td>
<td>0.74</td>
<td>0.3s</td>
</tr>
<tr>
<td>WOTR:exact</td>
<td>0.86</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WOTR:LD</td>
<td>0.92</td>
<td>0.84</td>
<td>0.80</td>
<td>31.6s</td>
</tr>
<tr>
<td>WOTR:DM</td>
<td>0.93</td>
<td>0.90</td>
<td>0.87</td>
<td>0.7s</td>
</tr>
<tr>
<td>FMP:exact</td>
<td>0.77</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FMP:LD</td>
<td>0.92</td>
<td>0.82</td>
<td>0.76</td>
<td>14.1s</td>
</tr>
<tr>
<td>FMP:DM</td>
<td>0.85</td>
<td>0.82</td>
<td>0.78</td>
<td>0.7s</td>
</tr>
</tbody>
</table>

Hosseini et al. (2020)
Coll Ardanuy et al. (2020b)
DeezyMatch can be applied for performing the following tasks:
- Fuzzy string matching
- Record linkage
- Candidate selection for entity linking systems
- Toponym matching

Table of contents
- Installation and setup
- Data and directory structure in tutorials
- Run DeezyMatch as a Python module or via command line
  - Quick tour
  - Train a new model
  - Fine-tune a pretrained model
  - Model inference
  - Generate query and candidate vectors
  - Candidate ranker and assembling vector representations
  - Candidate ranking on-the-fly
  - Tips / Suggestions on DeezyMatch functionalities
- Examples on how to run DeezyMatch
- Reproduce Fig. 2 of DeezyMatch’s paper, EMNLP2020
- How to cite DeezyMatch
- Credits

Installation

[GitHub Page](https://github.com/Living-with-machines/DeezyMatch)
A Flexible Deep Neural Network Approach to Fuzzy String Matching

DeezyMatch can be applied for performing the following tasks:

- Fuzzy string matching
- Record linkage
- Candidate selection for entity linking systems
- Toponym matching

Table of contents

- Installation and setup
- Data and directory structure in tutorials
- Run DeezyMatch as a Python module or via command line
  - Quick tour
  - Train a new model
  - Fine-tune a pretrained model
  - Model inference
  - Generate query and candidate vectors
  - Candidate ranker and assembling vector representations
  - Candidate ranking on-the-fly
  - Tips / Suggestions on DeezyMatch functionalities
- Examples on how to run DeezyMatch
- Reproduce Fig. 2 of DeezyMatch’s paper, EMNLP2020
- How to cite DeezyMatch
- Credits

Reproduce Fig. 2 of DeezyMatch's paper

The three notebooks in this directory can be used to reproduce Fig. 2 of DeezyMatch's paper:


- Fig2_EMNLP.training.ipynb: train and fine-tune a set of pair classifiers.
- Fig2_EMNLP.inference.ipynb: model inference using the models trained in the Fig2_EMNLP.training.ipynb notebook.
- Fig2_EMNLP.plot_results.ipynb: plots the results of model inference done in the Fig2_EMNLP.inference.ipynb notebook.
Current work  
(very early stage!)

Linking a directory of over 12k train stations to Wikidata using DeezyMatch.

Evolution of stations between 1800 and 1900.

Stations are colored by the first company operating the line.
Lessons learned
How to brainstorm ideas together

- HypGen: hypothesis generation group
- IdeasLab
- NLP reading group
- Computer vision for digital heritage interest group
- Humanities & data science discussion group
How to embed best RSE practices

- Offering git-flow overviews
- Being available for informal support (Code & Coffe)
- Having milestones independent from conference deadlines
- Having regular stand-up meetings
- Coding together and reviewing each other’s code (a lot)
# How to recognise all contributions

<table>
<thead>
<tr>
<th>Conceptualization</th>
<th>Methodology</th>
<th>Implementation</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Reproducibility</th>
<th>Interpretation</th>
<th>Historical Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kasra Hosseini, Federico Nanni</td>
<td>Kaspar Beelen, Mariona Coll Ardanuy, Katherine McDonough, Daniel CS Wilson, Ruth Ahnert, Jon Lawrence, Giorgia Tolfo</td>
<td>Daniel CS Wilson, Katherine McDonough, Kaspar Beelen, Jon Lawrence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Curation</th>
<th>Annotation</th>
<th>Writing and Editing</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supervision</th>
<th>Project Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barbara McGillivray, Ruth Ahnert</td>
<td>Barbara McGillivray, Ruth Ahnert, Mariona Coll Ardanuy</td>
</tr>
</tbody>
</table>

To know more: [https://livingwithmachines.ac.uk/highlighting-authors-contributions-and-interdisciplinary-collaborations-in-living-with-machines/](https://livingwithmachines.ac.uk/highlighting-authors-contributions-and-interdisciplinary-collaborations-in-living-with-machines/)
Thank you! Questions?
Finding Machines with a Dictionary
Finding Machines with a Dictionary

The Goals

**Overarching aim of the project:**
- Study the language of mechanisation

**Specific NLP research question:**
- **Where** do the machines live?
- Or: How to **define** machines and **detect** their presence in historical documents?

**General NLP task:**
- how to trace the manifestation of **concept** X, Y, Z in a **time-sensitive** manner?
Finding Machines with a Dictionary

*The Problem*

**Problem**: find mentions of “machines” (the token as well as the concept)

**Solution(?)**: Exploit information and structure of the *Oxford English Dictionary and Thesaurus* to algorithmically detect mentions of machines in text
Finding Machines with a Dictionary
Squeezing information from dictionaries

**Problem:** find mentions of “machines” (the token as well as the concept)

**Solution:** Exploit information and structure of the Oxford English Dictionary and Thesaurus) to algorithmically detect mentions of machines in text
Finding Machines with a Dictionary
Exploiting sense level information

Example for lemma id: machine_nn01

Sense 1:
- **Sense id**: machine_nn01-38476096
- **Definition**: “figurative. A living being considered to move or act automatically or mechanically …”
- **Quotation**: {id: ..., text: “… force men and women and children to degrade themselves into machines as wage-slaves”, year: 1910, etc.}
- **Semantic class**: [{'1', '8835', '25507', '29189'}]

Sense 2:
- **Sense id**: machine_nn01-XXXXXXXX

etc.
Finding Machines with a Dictionary
Exploiting sense level information

Example for lemma id: machine_nn01

Sense 1:
- **Sense id**: machine_nn01-38476096
- **Definition**: “figurative. A living being considered to move or act automatically or mechanically …”
- **Quotation**: {id: ..., text: “… force men and women and children to degrade themselves into machines as wage-slaves”, year: 1910, etc.}
- **Semantic class**: [['1', '8835', '25507', '29189']]

Sense 2:
- **Sense id**: machine_nn01-XXXXXXXX

etc.
Finding Machines with a Dictionary

Exploiting thesaurus structure

- **lemma**: machine_nn01
- **senses**: machine_nn01-384x ... machine_nn01-384x ... machine_nn01-384x
- **synonyms**: ... car_nn01-384x ... engine_nn01-384x ...
- **siblings and descendants**: ... wheel_nn01-384x ...
Finding Machines with a Dictionary
Exploiting thesaurus structure

lemma

senses

synonyms

siblings and descendants
Finding Machines with a Dictionary

Task Definition: Defining the concept

Input:
- A query lemma $L$ with $Q$ query senses
- A (set of) seed sense(s) $S \subseteq Q$
- A set of rules for expansion $R$

$R \subseteq \{\text{seed, synonym, sibling, descendant}\}$

$L, S, R$ Returns $C$
- A set of senses related to $S$, which we think of as representing the “concept”
Finding Machines with a Dictionary

Expanding the set of senses

In: \{machine\_nn01, \{machine\_nn01\-384y\}, synonyms\}

Out: \{machine\_nn01\-384y, locomotive\_nn01\-392o, engine\_nn01\-93y, ...\}

-> these are labelled 1, the remainder 0
Finding Machines with a Dictionary
Expanding the quotations

[They sell sewing-**machines**., 1889, machine_nn01-384y, 1]
[The **locomotive** was moving fast., 1860, locomotive_nn01-320x, 1]
...
[She walks like a **machine**., 1904, machine_nn01-394y, 0]
[He works as a **boiler**, 1854, boiler-nn01-54y, 0]
Finding Machines with a Dictionary

Expanding the quotations

Experiments with binary classification:
Baseline (adaptation of Hu et al. 2019)

- For all senses \( s \) in \( C \) (produced by \( \{Q,S,R\} \))
  - Label associated quotations as 1; Rest as 0
- For each labelled quotations (text with target words)
  - E. g. … (force men and women and children to degrade themselves into machines as wage-slaves, 1)
  - Obtain contextualized vector of target word, and average vectors by category \( (v_0, v_1) \)
    - “Concept embedding” for C and not-C
  - For each word \( w \) in sent take \( \arg\max(\text{sim}(v_0, w(v)), \text{sim}(v_1, w(v))) \)
Finding Machines with a Dictionary

Expanding the set of senses

They sell sewing-**machines**.
The **locomotive** was moving fast.

Class 1

She walks like a **machine**.
He works as a **boiler**.
Finding Machines with a Dictionary
Expanding the set of senses

They sell sewing-\textcolor{blue}{machines}.  
The \textcolor{red}{locomotive} was moving fast.

Class 1

I bought a flying \textcolor{blue}{machine}

Class 0

She walks like a \textcolor{blue}{machine}.  
He works as a \textcolor{red}{boiler}.  

\begin{tabular}{|c|c|c|c|}
\hline
Class 1 & Class 0 & Class 1 & Class 0 \\
\hline
0.54 & 0.23 & 0.98 & 0.45 \\
0.97 & 0.23 & 0.45 & 0.97 \\
\ldots & \ldots & \ldots & \ldots \\
\hline
\end{tabular}
# Finding Machines with a Dictionary

*Expanding the set of senses*

<table>
<thead>
<tr>
<th>Class 1</th>
<th>Class 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>They sell sewing-\textbf{machines}, 1889</td>
<td>She walks like a \textbf{machine}, 1904</td>
</tr>
<tr>
<td>The \textbf{locomotive} was moving fast., 1860</td>
<td>He works as a \textbf{boiler}, 1854</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vector class 1</th>
<th>Vector class 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.97 0.23 ... 0.45</td>
<td>0.97 0.23 ... 0.45</td>
</tr>
<tr>
<td>0.54 0.23 ... 0.98</td>
<td>0.54 0.25 ... 0.98</td>
</tr>
</tbody>
</table>
Finding Machines with a Dictionary
Expanding the set of senses

They sell sewing- **machines**, 1889

The **locomotive** was moving fast., 1860

Class 1

I bought a flying **machine**, 1880

Class 0

She walks like a **machine**, 1904

He works as a **boiler**, 1854
Finding Machines with a Dictionary

Improve on baseline by making disambiguation time sensitive

- **Weighted** or selective averaging for constructing the concept embedding (quotations closer in time have more weight etc)

- **Adapt BERT** for historical WSD
  - Fine-tune BERT-models on historical data
  - Adapt pre-training task (SenseBERT), fine-tune with additional information (GLOSSBERT)

- **Adapt disambiguation step** (Nearest Neighbour, Stack FC layer, etc.)
Questions
DeezyMatch

Example timeline

**Week(s)**

- **Start**
  - Hypothesis, initial questions

- **Turn ideas into DS / SE (sub-)tasks**

- **EDA**
  - Exploratory Data Analysis
  - Design flowchart

- **Hack week(s)**
  - Exploring available datasets / libraries / methods
  - Identify limitations of existing methods or datasets
  - Do we need new method(s) for our tasks?
  - Collect more data?
  - Baselines

**Month(s)**

- **Set up infrastructure**
  - Repositories (e.g., GitHub), project board for planning / sprint meetings
  - Public repos from the beginning (if at all possible)
  - Create DB

- **Implement new methods**
  - Test CL reproducibility, develop a new library?

- **Submit paper**
  - Make repo public?
    - (documentation, env, ...)

- **Results and analysis**
  - Write paper

- **Maintain**
  - Other processes
  - Address any additional functions