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

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:
- 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

Claire Austin
British Library

Dr Kaspar Beelen
Research Associate

Dr Mariona Coll Ardanuy
Research Associate

Dr Kasra Hosseini
Research Data Scientist

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 Piza
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) Month(s) year(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**

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

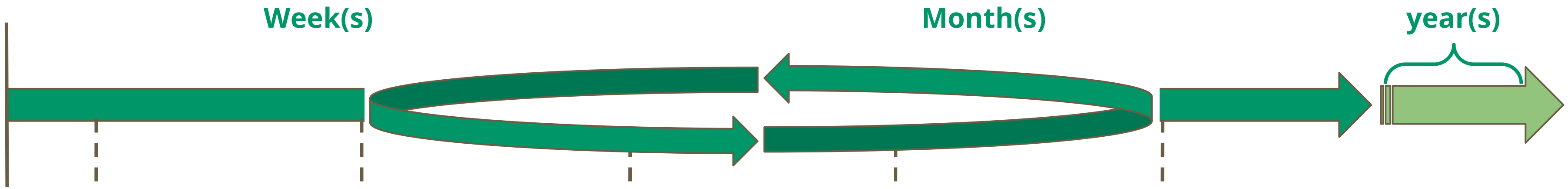
Make repo **public**? (documentation, env, …)

Maintenance

Other users, projects?

Address issues, add more functionalities, …

Example timeline



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**

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, ...

Deadline is approaching!!!

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

Detecting living machines: motivation

- 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 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

19thC Machines animacy dataset

Annotation

Annotation was challenging, even for domain experts.

"No, no, to her mother poor Fraulein was not a woman, a heart, a soul; she was just a <u>machine</u>."

Into an Unknown World. A novel, 1897, J.S. Winter

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.

19thC Machines animacy dataset (III)

- 593 sentences: 201 animate/292 inanimate expressions
- Krippendorff's α=0.74 on animacy, α=0.50 on humanness.
- Rich in atypical animacy.

Target	Sentence	Animacy	Humanness
engine	In December, the first steam fire **engine** was received, and tried on the shore of Lake Monona, with one thousand feet of hose.	0	0
engine	It was not necessary for Jakie to slow down in order to allow the wild **engine** to come up with him; she was coming up at every revolution of her wheels.	1	1
locomotive	Nearly a generation had been strangely neglected to grow up un-Americanized, and the private adventurer and the **locomotive** were the untechnical missionaries to open a way for the common school.	1	1
machine	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 **machine**, held in by the brake, squeaked a pitiful warning.	1	1
machines	Our servants, like mere **machines**, move on their mercenary track without feeling.	1	0
machinery	We have everywhere water power to any desirable extent, suitable for propelling all kinds of **machinery**.	0	0

Approach in a nutshell

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

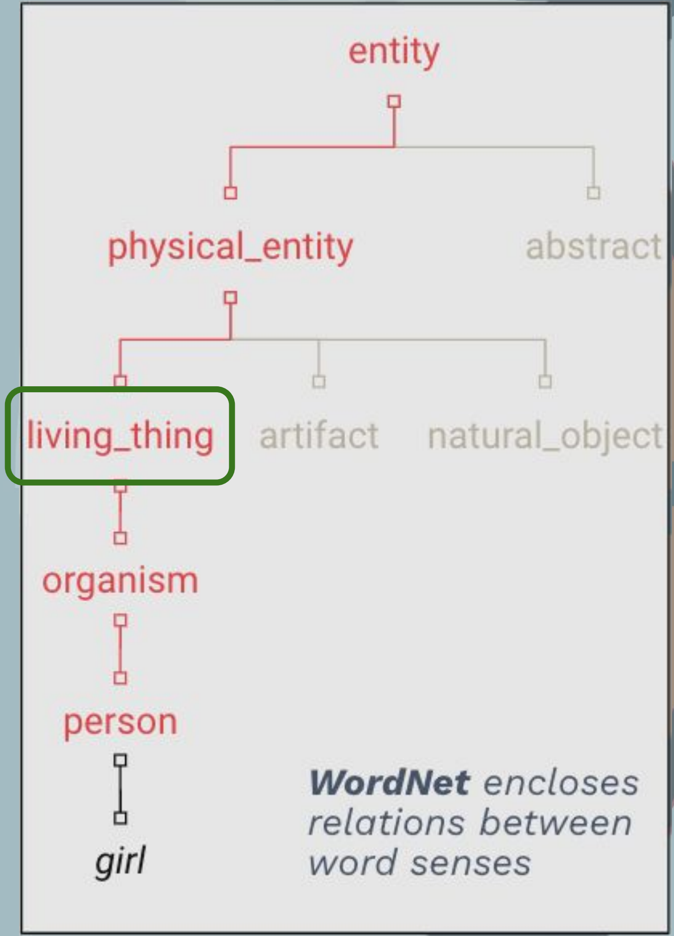
Joy and sorrow – life and death,
wrote the little [MASK] .

BERT, predict the missing word in the sentence:

girl	8.1641
man	8.0409
prince	7.4537
one	7.2818
boy	6.9801
princess	6.6766
bird	6.6638
voice	6.6378
lady	6.5472
angel	6.4725
wolf	6.4654
queen	6.3818
witch	6.3068
king	6.2712
sister	6.1635
brother	6.1291

Determining animacy

- <u>Assumption:</u> 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...

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

Experiments: baselines

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

Results

	Stories				*19thC Machines*			
	Precision	Recall	F-Score	Map	Precision	Recall	F-Score	Map
Most frequent class	0.31	0.5	0.383	0.623	0.336	0.5	0.402	0.318
SVM TFIDF: targetExp	0.911	0.893	0.902	0.928	0.696	0.713	0.704	0.474
SVM WordEmb: targetExp	0.927	0.919	0.923	0.954	0.694	0.711	0.702	0.499
BERTClassifier: targetExp	0.951	**0.948**	0.949	**0.985**	0.698	0.715	0.706	0.51
SVM TFIDF: targetExp + ctxt	0.734	0.739	0.737	0.859	0.688	0.71	0.699	0.651
SVM WordEmb: targetExp + ctxt	0.758	0.742	0.75	0.876	0.728	0.531	0.614	0.481
BERTClassifier: targetExp + ctxt	0.931	0.926	0.929	0.978	0.695	0.721	0.708	0.721
SVM TFIDF: maskedExp + ctxt	0.674	0.677	0.675	0.804	0.592	0.6	0.596	0.498
SVM WordEmb: maskedExp + ctxt	0.674	0.678	0.676	0.809	0.518	0.52	0.519	0.339
BERTClassifier: maskedExp + ctxt	0.855	0.852	0.854	0.951	0.687	0.696	0.692	0.603
SeqModel: LSTM	**0.952**	**0.948**	**0.95**	0.949	0.697	0.719	0.708	0.482
MaskPredict: BERT-base	0.739	0.703	0.72	0.848	0.719	0.742	0.73	0.74
MaskPredict: BERT-base +ctxt	0.839	0.774	0.806	0.892	0.758	**0.778**	0.768	**0.795**
MaskPredict: fit19thBERT +ctxt	–	–	–	–	0.758	0.775	0.766	0.777
MaskPredict: early19thBERT +ctxt	–	–	–	–	**0.799**	0.773	**0.786**	0.784

Results

	Stories				*19thC Machines*			
	Precision	Recall	F-Score	Map	Precision	Recall	F-Score	Map
Most frequent class	0.31	0.5	0.383	0.623	0.336	0.5	0.402	0.318
SVM TFIDF: targetExp	0.911	0.893	0.902	0.928	0.696	0.713	0.704	0.474
SVM WordEmb: targetExp	0.927	0.919	0.923	0.954	0.694	0.711	0.702	0.499
BERTClassifier: targetExp	0.951	**0.948**	0.949	**0.985**	0.698	0.715	0.706	0.51
SVM TFIDF: targetExp + ctxt	0.734	0.739	0.737	0.859	0.688	0.71	0.699	0.651
SVM WordEmb: targetExp + ctxt	0.758	0.742	0.75	0.876	0.728	0.531	0.614	0.481
BERTClassifier: targetExp + ctxt	0.931	0.926	0.929	0.978	0.695	0.721	0.708	0.721
SVM TFIDF: maskedExp + ctxt	0.674	0.677	0.675	0.804	0.592	0.6	0.596	0.498
SVM WordEmb: maskedExp + ctxt	0.674	0.678	0.676	0.809	0.518	0.52	0.519	0.339
BERTClassifier: maskedExp + ctxt	0.855	0.852	0.854	0.951	0.687	0.696	0.692	0.603
SeqModel: LSTM	**0.952**	**0.948**	**0.95**	0.949	0.697	0.719	0.708	0.482
MaskPredict: BERT-base	0.739	0.703	0.72	0.848	0.719	0.742	0.73	0.74
MaskPredict: BERT-base +ctxt	0.839	0.774	0.806	0.892	0.758	**0.778**	0.768	**0.795**
MaskPredict: fit19thBERT +ctxt	–	–	–	–	0.758	0.775	0.766	0.777
MaskPredict: early19thBERT +ctxt	–	–	–	–	**0.799**	0.773	**0.786**	0.784

Results

	Stories				*19thC Machines*			
	Precision	Recall	F-Score	Map	Precision	Recall	F-Score	Map
Most frequent class	0.31	0.5	0.383	0.623	0.336	0.5	0.402	0.318
SVM TFIDF: targetExp	0.911	0.893	0.902	0.928	0.696	0.713	0.704	0.474
SVM WordEmb: targetExp	0.927	0.919	0.923	0.954	0.694	0.711	0.702	0.499
BERTClassifier: targetExp	0.951	**0.948**	0.949	**0.985**	0.698	0.715	0.706	0.51
SVM TFIDF: targetExp + ctxt	0.734	0.739	0.737	0.859	0.688	0.71	0.699	0.651
SVM WordEmb: targetExp + ctxt	0.758	0.742	0.75	0.876	0.728	0.531	0.614	0.481
BERTClassifier: targetExp + ctxt	0.931	0.926	0.929	0.978	0.695	0.721	0.708	0.721
SVM TFIDF: maskedExp + ctxt	0.674	0.677	0.675	0.804	0.592	0.6	0.596	0.498
SVM WordEmb: maskedExp + ctxt	0.674	0.678	0.676	0.809	0.518	0.52	0.519	0.339
BERTClassifier: maskedExp + ctxt	0.855	0.852	0.854	0.951	0.687	0.696	0.692	0.603
SeqModel: LSTM	**0.952**	**0.948**	**0.95**	0.949	0.697	0.719	0.708	0.482
MaskPredict: BERT-base	0.739	0.703	0.72	0.848	0.719	0.742	0.73	0.74
MaskPredict: BERT-base +ctxt	0.839	0.774	0.806	0.892	0.758	**0.778**	0.768	**0.795**
MaskPredict: fit19thBERT +ctxt	–	–	–	–	0.758	0.775	0.766	0.777
MaskPredict: early19thBERT +ctxt	–	–	–	–	**0.799**	0.773	**0.786**	0.784

A Reproducible Experimental Setting

Living Machines:
A Study of Atypical Animacy

License MIT

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

Table of contents

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
ASHTONCNDER-LYNE
Ashton-under-lyne
Ashtonunder-Lyne
ASHTON-UXDER-LYNE
Ashton-cnder-Ltne
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

Ashton-under-Lyne (Q659803)

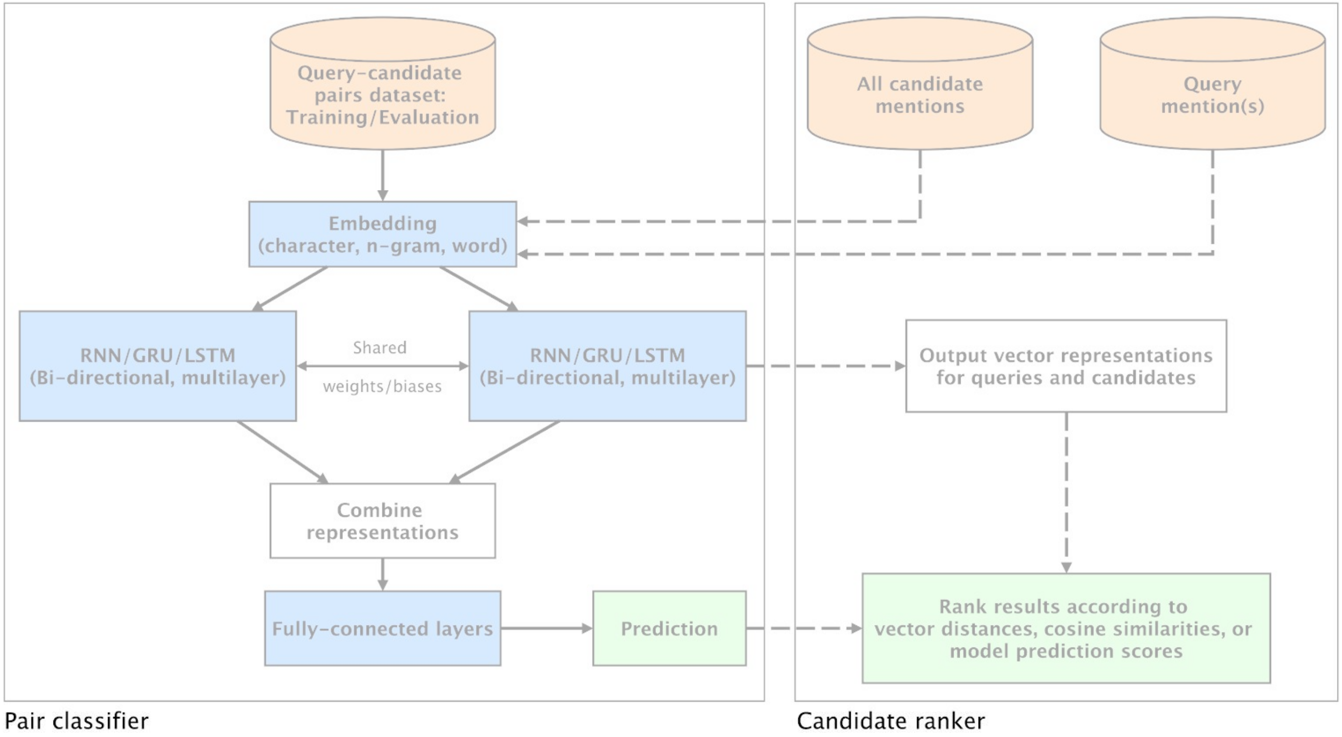
market town in the Metropolitan Borough of Tameside, Greater Manchester, England

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

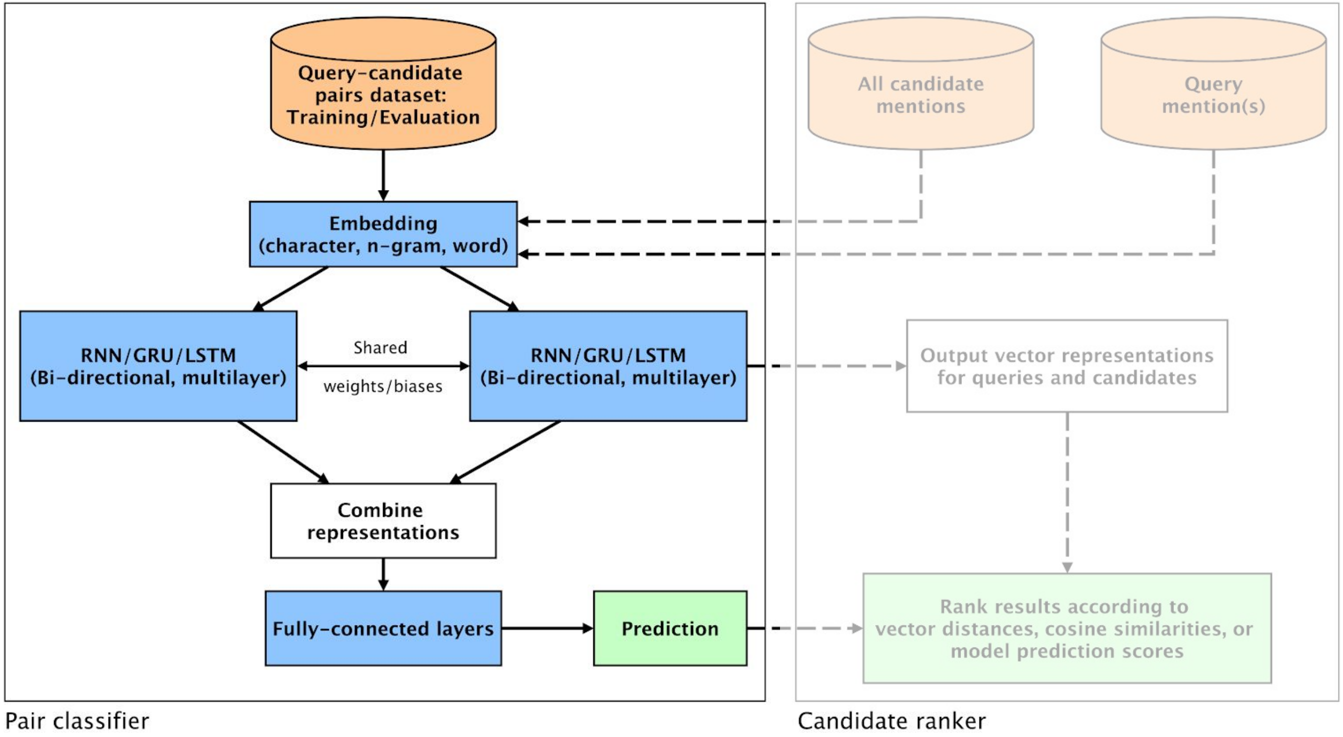
A flexible deep learning approach to fuzzy string matching and candidate ranking.



Hosseini et al. (2020)

DeezyMatch: architecture

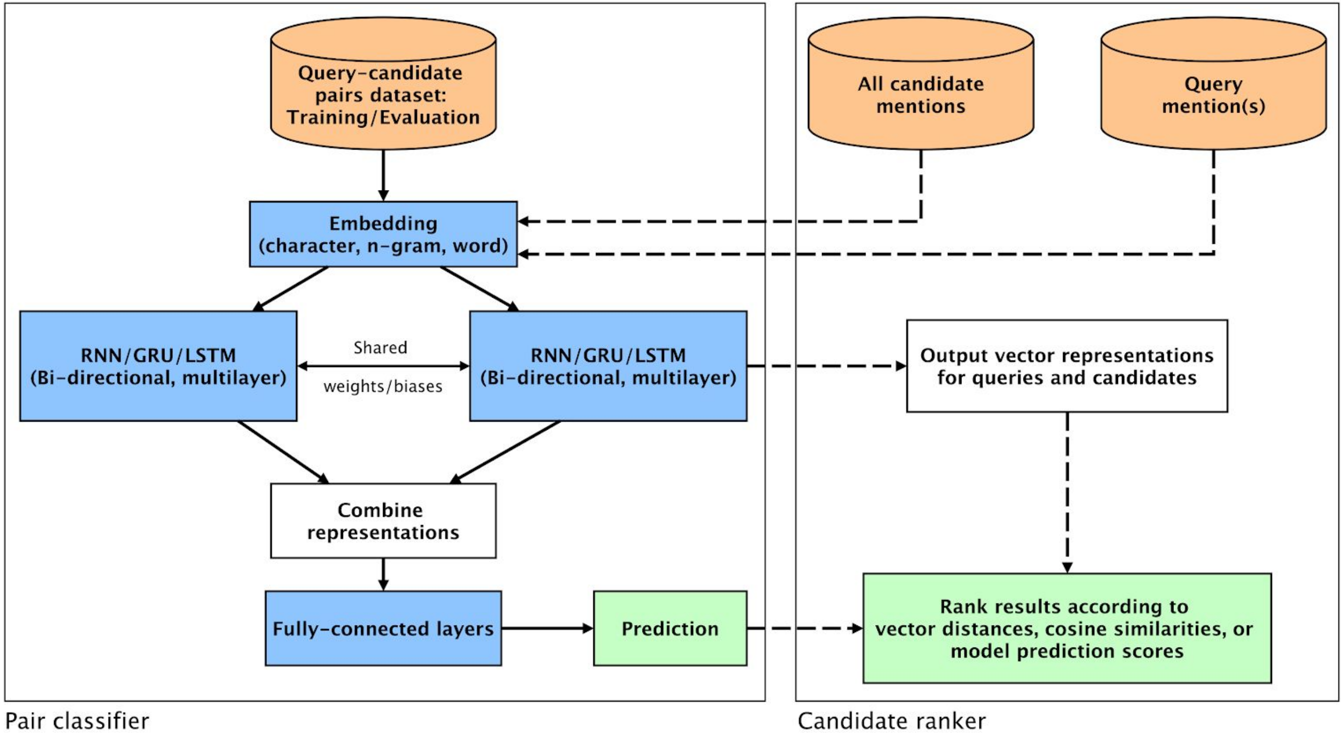
A flexible deep learning approach to fuzzy string matching and candidate ranking.



Hosseini et al. (2020)

DeezyMatch: architecture

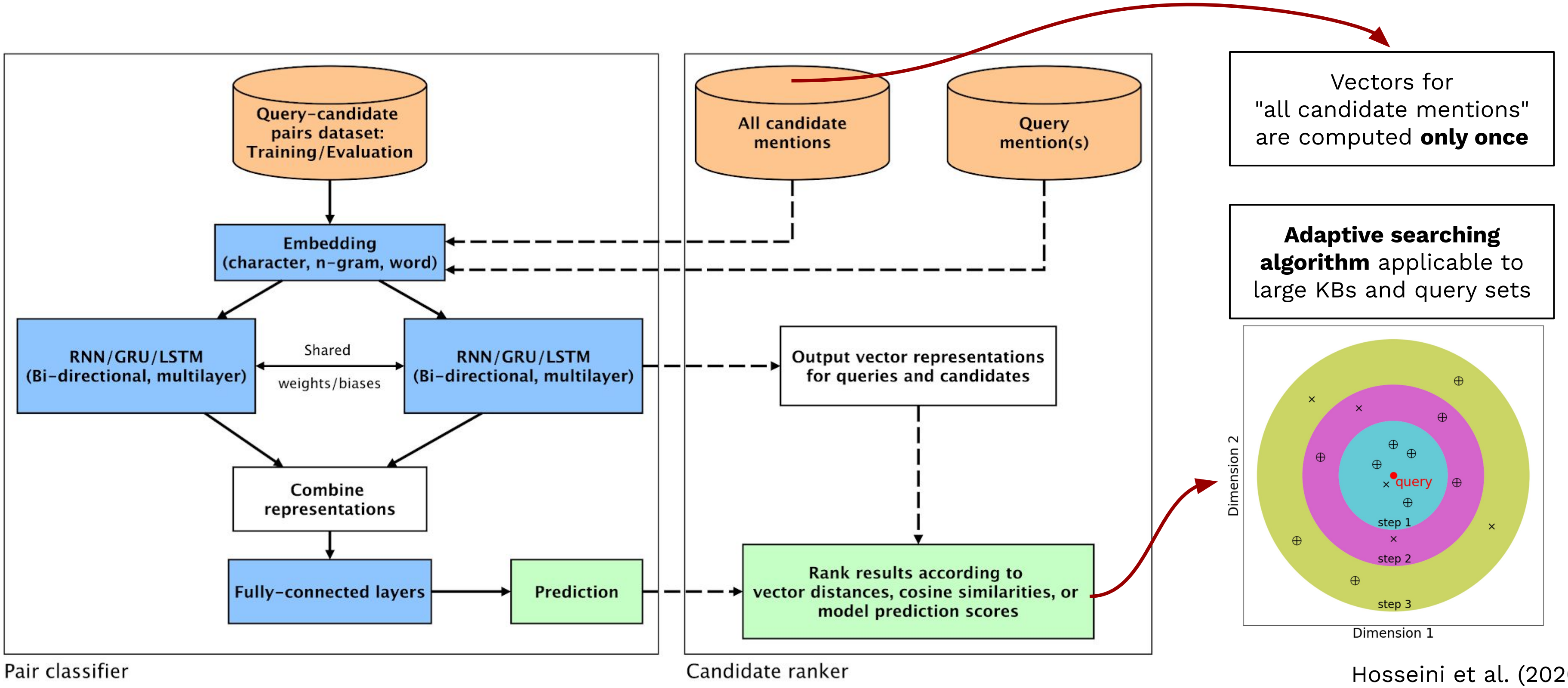
A flexible deep learning approach to fuzzy string matching and candidate ranking.



Hosseini et al. (2020)

DeezyMatch: architecture

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



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.

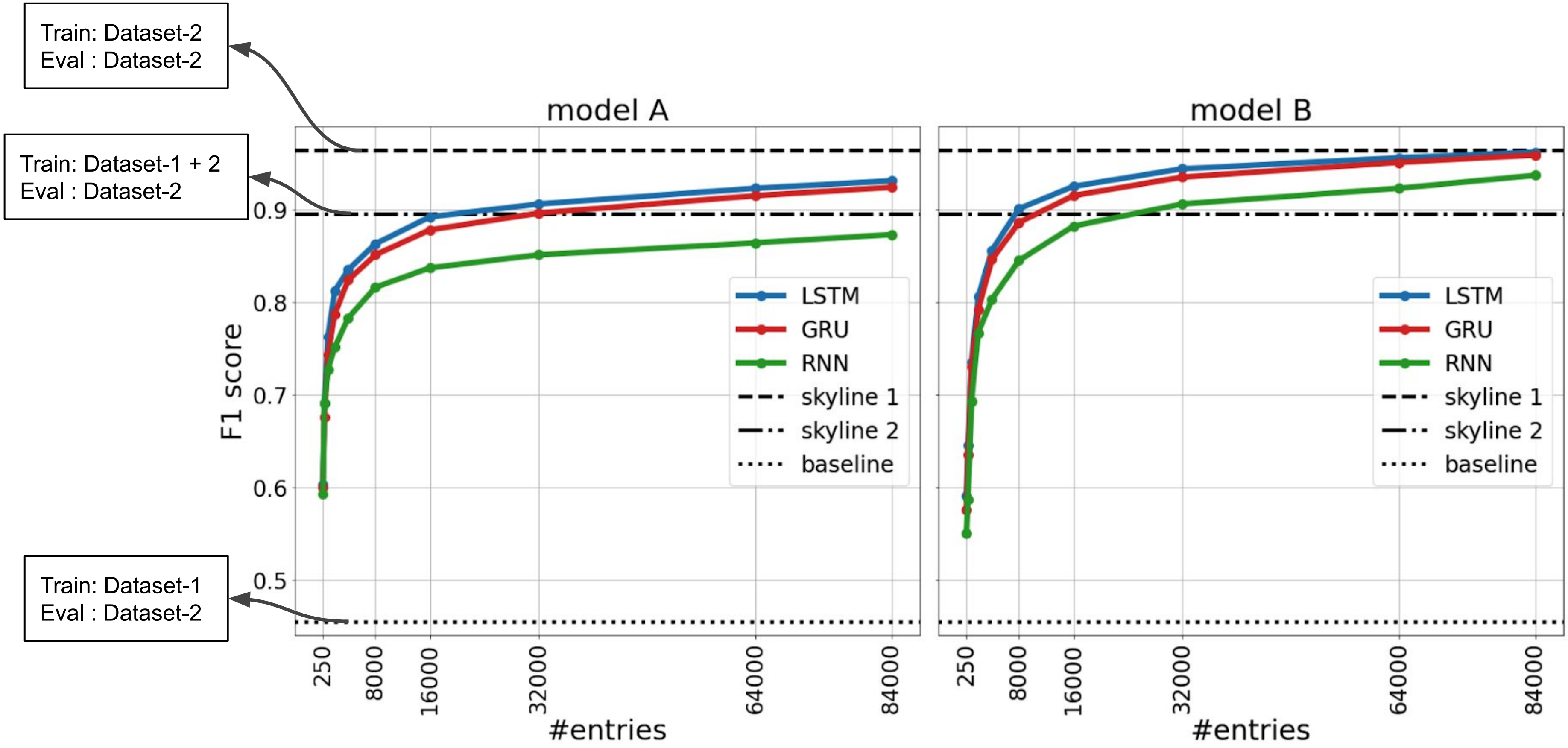
- User can change the architecture (RNN, GRU or LSTM), hyperparameters and preprocessing steps via input file.

```
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)



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.

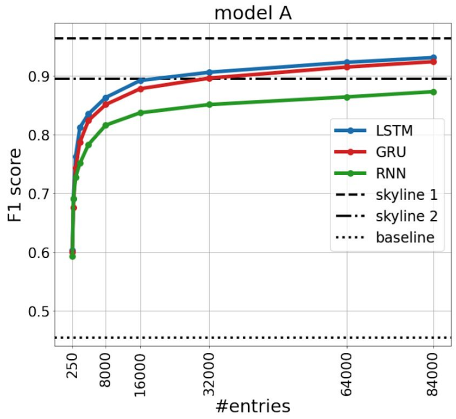
- Fine-tuning a pretrained model; transfer learning.

- Extensive documentation:
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)
```



Hosseini et al. (2020)

DeezyMatch: performance

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

	Santos	WG:en	OCR
LevDam	0.70	0.74	0.76
Santos et al. (2018a)	0.82	0.92	0.95
DeezyMatch	0.89	0.94	0.95

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

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

Hosseini et al. (2020)
Coll Ardanuy et al. (2020b)

README.md

Deezy Match

A Flexible Deep Neural Network Approach to Fuzzy String Matching

pypi v1.2.3 | License MIT | launch binder | Continuous integration passing

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

https://github.com/Living-with-machines/DeezyMatch

Hosseini et al. (2020)
Coll Ardanuy et al. (2020b)

README.md

A Flexible Deep Neural Network Approach to Fuzzy String Matching

pypi v1.2.3 | License MIT | launch binder | Continuous integration passing

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

master | DeezyMatch / figs / EMNLP2020_figures / fig2 / | Go to file | Add file | ···

kasra-hosseini Add information about models to the notebook | cfdcb58 on 10 Nov 2020 | History

..		
inputs	* Move notebooks to a new directory: figs/EMNLP2020_figures/fig2	5 months ago
Fig2_EMNLP_inference.ipynb	Add information about models to the notebook	5 months ago
Fig2_EMNLP_plot_results.ipynb	Add information about models to the notebook	5 months ago
Fig2_EMNLP_training.ipynb	Add information about models to the notebook	5 months ago
README.md	* Move notebooks to a new directory: figs/EMNLP2020_figures/fig2	5 months ago

README.md

Reproduce Fig. 2 of DeezyMatch's paper

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

```
Hosseini, Nanni and Coll Ardanuy (2020), DeezyMatch: A Flexible Deep Learning Approach to Fuzzy String Matching, EMNLP: Sy
```

- `Fig2_EMNLP_training.ipynb` : train and fine-tune a suit of pair classifiers.
- `Fig2_EMNLP_inference.ipynb` : model inference using the models trained in the `Fig2_EMNLP_training.ipyn` notebook.
- `Fig2_EMNLP_plot_results.ipynb` : plot the results of model inference done in the `Fig2_EMNLP_inference` notebook.

https://github.com/Living-with-machines/DeezyMatch

Hosseini et al. (2020)
Coll Ardanuy et al. (2020b)

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

Conceptualization
Mariona Coll Ardanuy[1,5]
Daniel CS Wilson[1,5]

Methodology
Mariona Coll Ardanuy
Federico Nanni[1]
Kasra Hosseini[1]

Implementation
Federico Nanni
Kasra Hosseini
Mariona Coll Ardanuy
Kaspar Beelen[1,5]

Reproducibility
Kasra Hosseini
Federico Nanni

Interpretation
Kaspar Beelen
Mariona Coll Ardanuy
Katherine McDonough[1,5]
Daniel CS Wilson
Ruth Ahnert[5]
Jon Lawrence[4]
Giorgia Tolfo[2]

Historical Analysis
Daniel CS Wilson
Katherine McDonough
Kaspar Beelen
Jon Lawrence

Data Curation
Kaspar Beelen
Mariona Coll Ardanuy
Federico Nanni
Giorgia Tolfo

Annotation
Giorgia Tolfo
Ruth Ahnert
Kaspar Beelen
Mariona Coll Ardanuy
Jon Lawrence
Katherine McDonough
Federico Nanni
Daniel CS Wilson

Writing and Editing
Mariona Coll Ardanuy
Federico Nanni
Ruth Ahnert
Kaspar Beelen
Kasra Hosseini
Jon Lawrence
Katherine McDonough
Barbara McGillivray[1,3]
Daniel CS Wilson

Supervision
Barbara McGillivray
Ruth Ahnert

Project Management
Barbara McGillivray
Ruth Ahnert
Mariona Coll Ardanuy

To know more: 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-XXXXXXX

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-XXXXXXX

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***

machine_nn01

lemma

senses

machine_nn01-384x

machine_nn01-394y

machine_nn01-384z

synonyms

car_nn01-494x

engine_nn01-385u

siblings and descendants

wheel_nn01-84x

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 ⊆ Q**
- A set of rules for expansion **R**

R ⊆ {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 argmax(sim(v_0, w(v)), sim(v_1, w(v)))

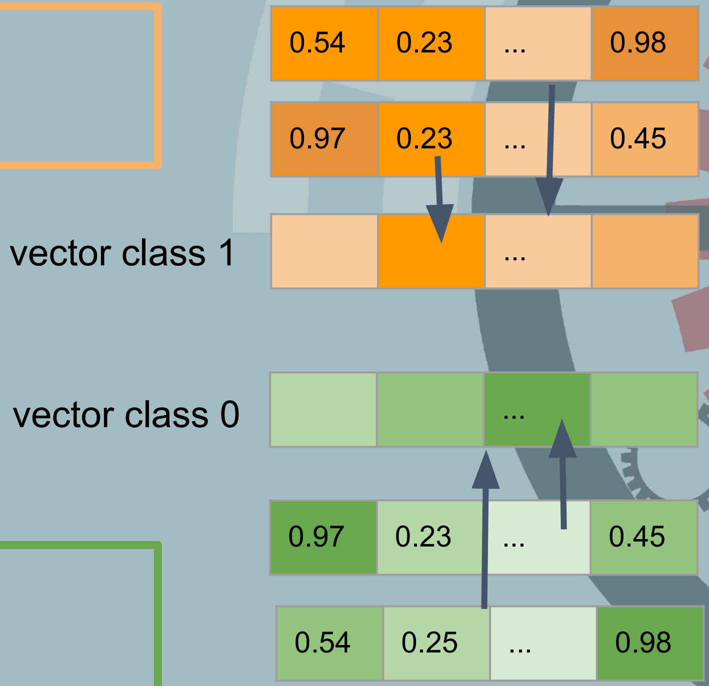
Finding Machines with a Dictionary

Expanding the set of senses

They sell sewing-**machines**.

The **locomotive** was moving fast.

Class 1



Class 0

She walks like a **machine**.

He works as a **boiler**.

Finding Machines with a Dictionary

Expanding the set of senses

They sell sewing-**machines**.

The **locomotive** was moving fast.

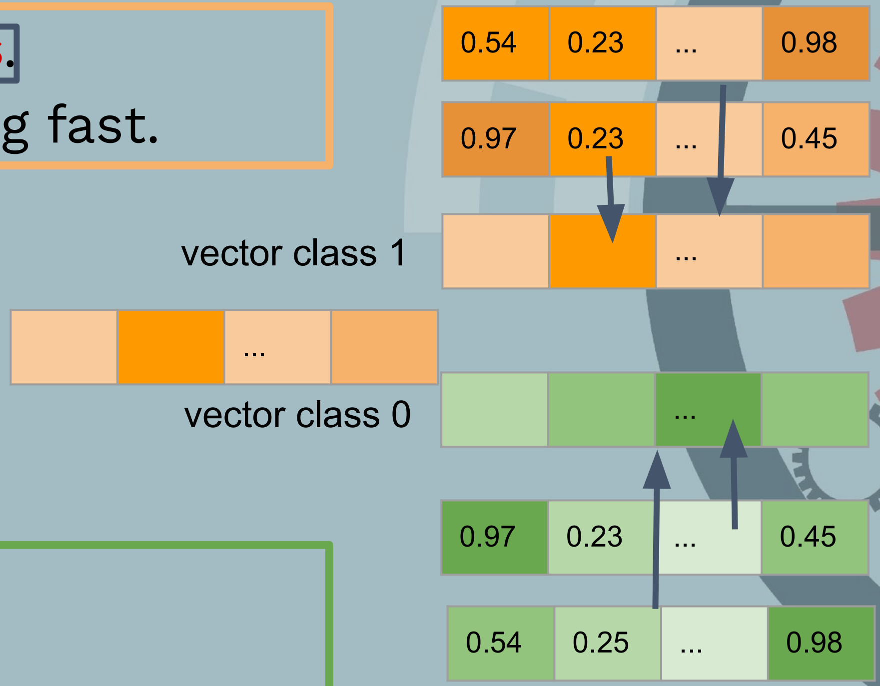
Class 1

I bought a flying **machine**

Class 0

She walks like a **machine**.

He works as a **boiler**.



Finding Machines with a Dictionary

Expanding the set of senses

They sell sewing-**machines**., **1889**

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

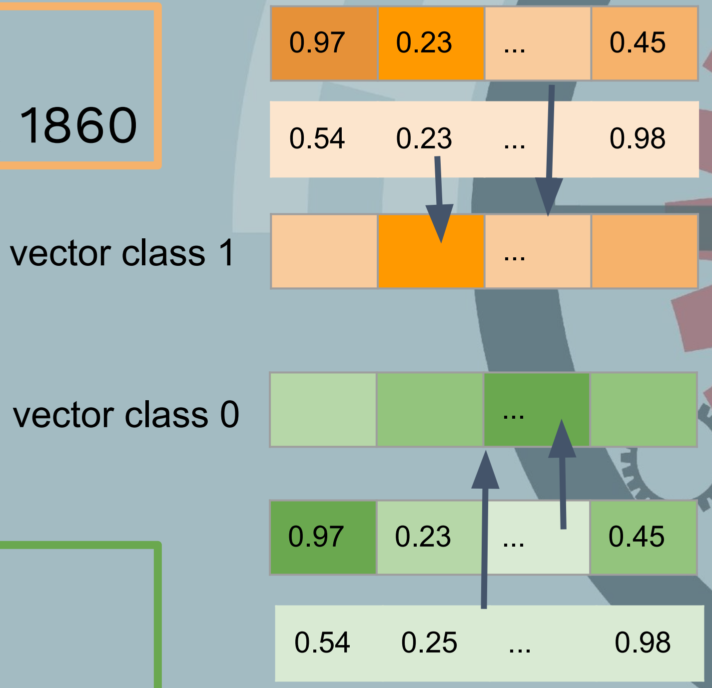
Class 1

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

Expanding the set of senses

They sell sewing-**machines**., **1889**

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

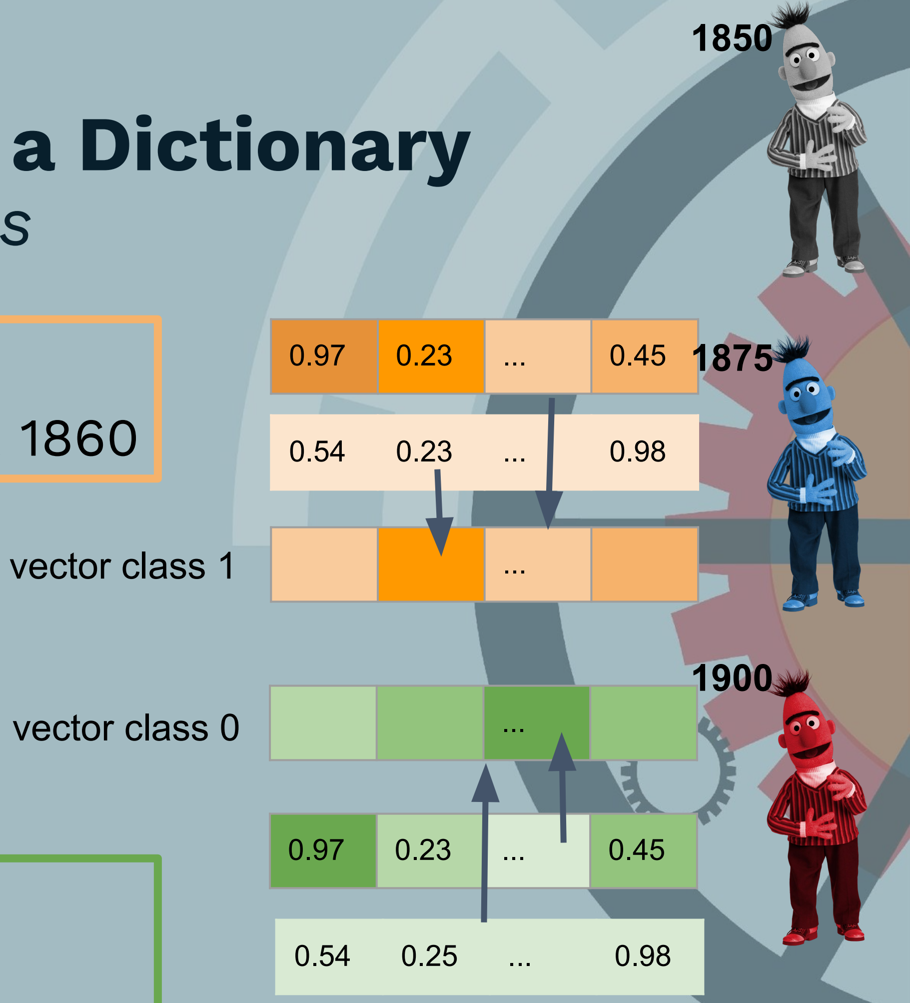
Class 1

I bought a flying **machine**., **1880**

Class 0

She walks like a **machine**., **1904**

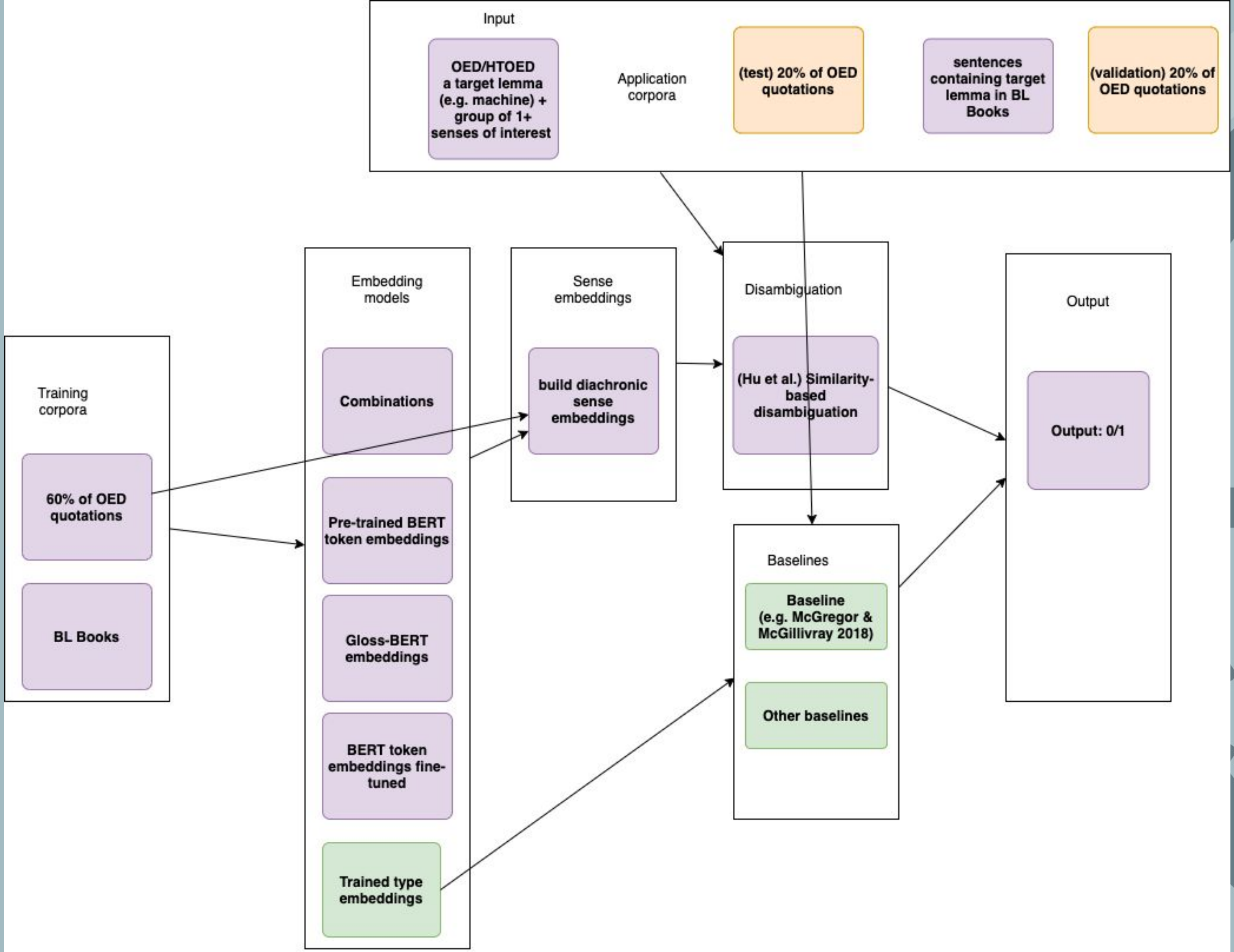
He works as a **boiler**., 1854



Finding Machines with a Dictionary

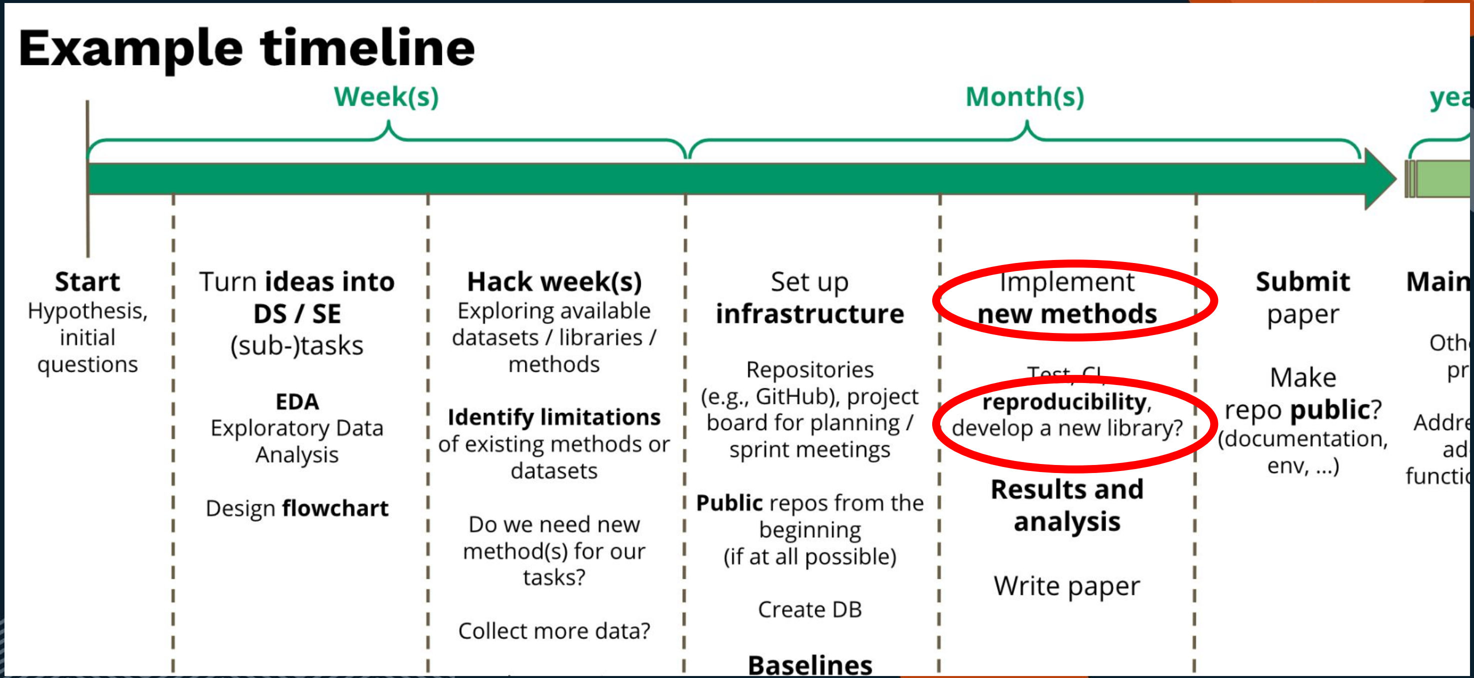
Improve on baseline by making disambiguation time sensitive

- **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.)


Input
OED/HTOED a target lemma (e.g. machine) + group of 1+ senses of interest
Application corpora
(test) 20% of OED quotations
sentences containing target lemma in BL Books
(validation) 20% of OED quotations
Training corpora
60% of OED quotations
BL Books
Embedding models
Combinations
Pre-trained BERT token embeddings
Gloss-BERT embeddings
BERT token embeddings fine-tuned
Trained type embeddings
Sense embeddings
build diachronic sense embeddings
Disambiguation
(Hu et al.) Similarity-based disambiguation
Baselines
Baseline (e.g. McGregor & McGillivray 2018)
Other baselines
Output
Output: 0/1

Questions

DeezyMatch


Example timeline
Week(s)
Month(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?
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
Implement new methods
Test, CI, reproducibility, develop a new library?
Results and analysis
Write paper
Submit paper
Make repo public? (documentation, env, ...)