Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

PAR4SIM – ADAPTIVE PARAPHRASING FOR TEXT SIMPLIFICATION

SEID MUHIE YIMAM & CHRIS BIEMANN

COLING 2018, SANTA FE, NEW MEXICO, USA

MOTIVATION

- Machine learning applications require to collect training data manually
- **Limitations**:
  - Expensive
  - Time consuming
  - Training/teaching of annotators/experts is needed
  - Concept drift
- **Suggestions**:
  - Embedded and adaptive model
    - Usage data as training dataset
  - Human-in-the-loop approaches
    - No separate annotation process
  - Personalized applications
    - No extra expert training is required
  - Continuous model adaptation
    - Model learns the changes over time

RESEARCH QUESTIONS

- How can an **adaptive paraphrase** ranking model be integrated into a **text simplification** writing aid tool?
- How can an **adaptive paraphrase** ranking model be **evaluated**?
- Can we **demonstrate** the **adaptivity** of the approach?

EXPERIMENTAL SETUP AND RESOURCES

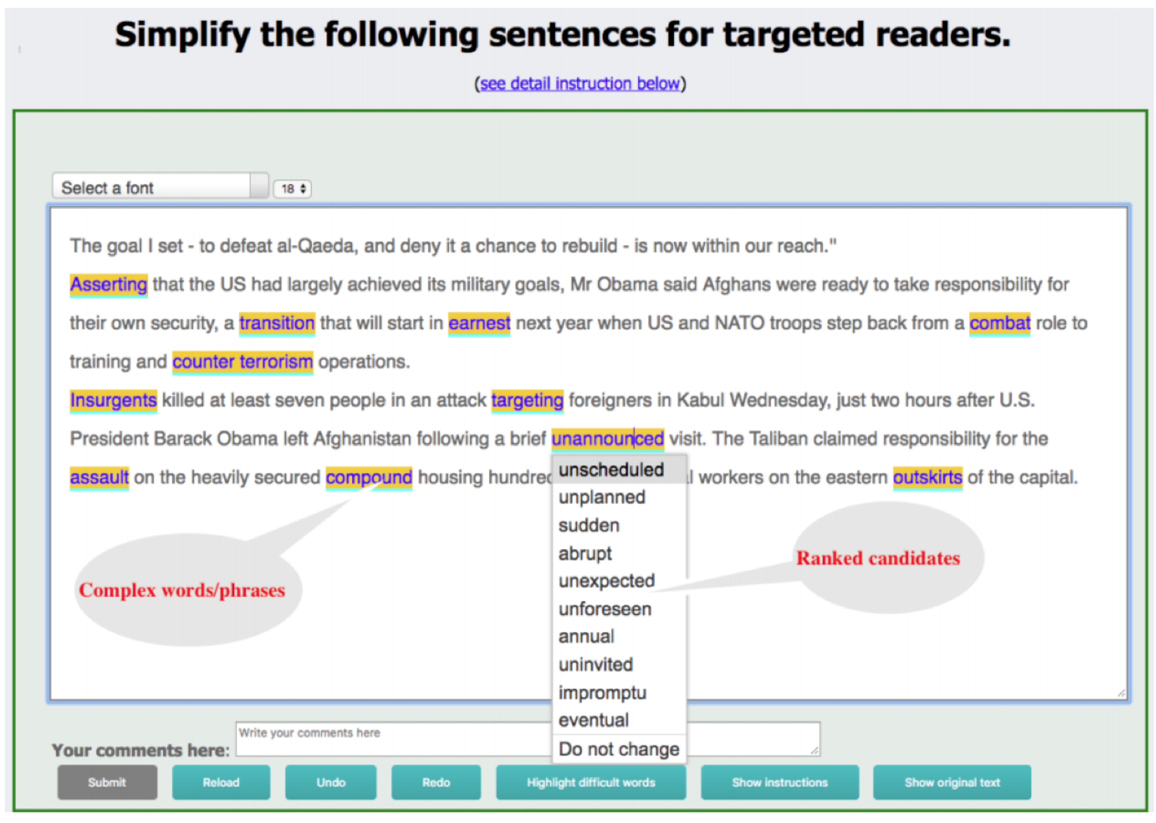
- Highlight words or phrases (complex phrases – CPs) to simplify a given text that is difficult to understand for particular readers such as **language learners, children** or **people with reading impairments**.
- Complex word identification (CWI) datasets from (Yimam et al. 2017) are used to highlight CPs for the text simplification system hosted on Amazon Mechanical Turk (Mturk).
- The adaptive paraphrase ranking system runs on our local servers, which communicate to the MTurk system via **external HITs**.

- **Candidates generation**
  - Lexical and distributional resources: WordNet and distributional thesaurus (Miller, 1995 and Biemann et al., 2013)
  - PPDB 2.0 and simple PPDB (Pavlick et al., 2015, Pavlick and Callison Burch, 2016)
  - Phrase2Vec: Phrase2Vec model (Mikolov et al., 2013) using English Wikipedia and the AQUAINT corpus of English news text (Graff, 2002).
- **Resources**:
  - A total of **10.8K training instances** are collected using **71** different **workers** from **3 countries**.

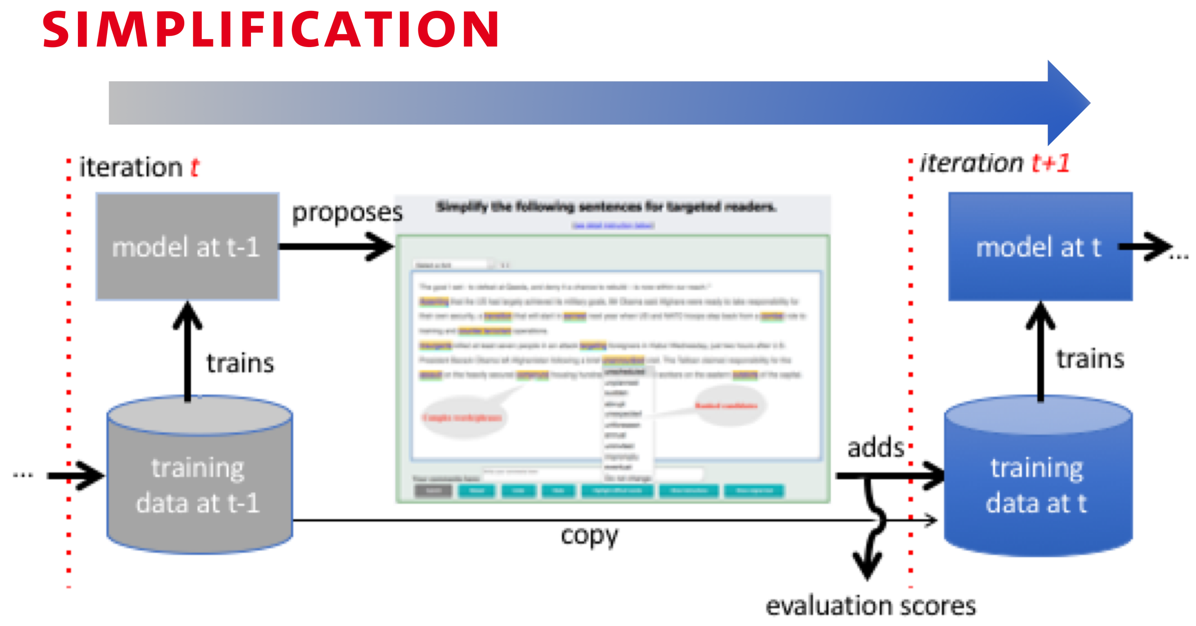
LEARNING-TO-RANK

- The system is trained by providing pairs of texts with complex phrases and candidates along their ideal rankings.
- **Ranklib** is used to build the learning and ranking models
- **LambdaMART** algorithm is selected to train the models.
- **Selected features**:
  - Frequency and length
  - Lexical and distributional thesaurus resources
  - PPDB 2.0 and simple PPDB
  - Word embeddings feature

USER INTERFACE OF PAR4SIM



ADAPTIVE MODELS FOR TEXT SIMPLIFICATION



- First iteration: use baseline language model ranking
- At iteration t, usage data from t-1 is used to build the model
- Training data is built in batches
- No overlap of data between iterations

INSTANCES OF TRAINING DATASET

Complex sentence: *Hajar said his cousin was not **affiliated** with any terrorist group.*	
Simplified sentence 1: *Hajar said his cousin was not **associated** with any terrorist group.*	→ 6
Simplified sentence 2: *Hajar said his cousin was not **merged** with any terrorist group.*	→ 2
Simplified sentence 3: *Hajar said his cousin was not **aligned** with any terrorist group.*	→ 1
Simplified sentence 4: *Hajar said his cousin was not **partnered** with any terrorist group.*	→ 1

Examples of usage data as training instances. Here affiliated is a complex phrase highlighted based on the CWI dataset and associated, merged, aligned, and partnered are the simpler options provided by 6, 2, 1, and 1 workers respectively.

RESULTS AND DISCUSSIONS

- Adaptive paraphrase ranking model effectively improves the performance of text simplification task.
- Domain adaption can be combined with adaptive machine learning
- Personalized NLP application is a way forward.
- User interface design is central for adaptive systems

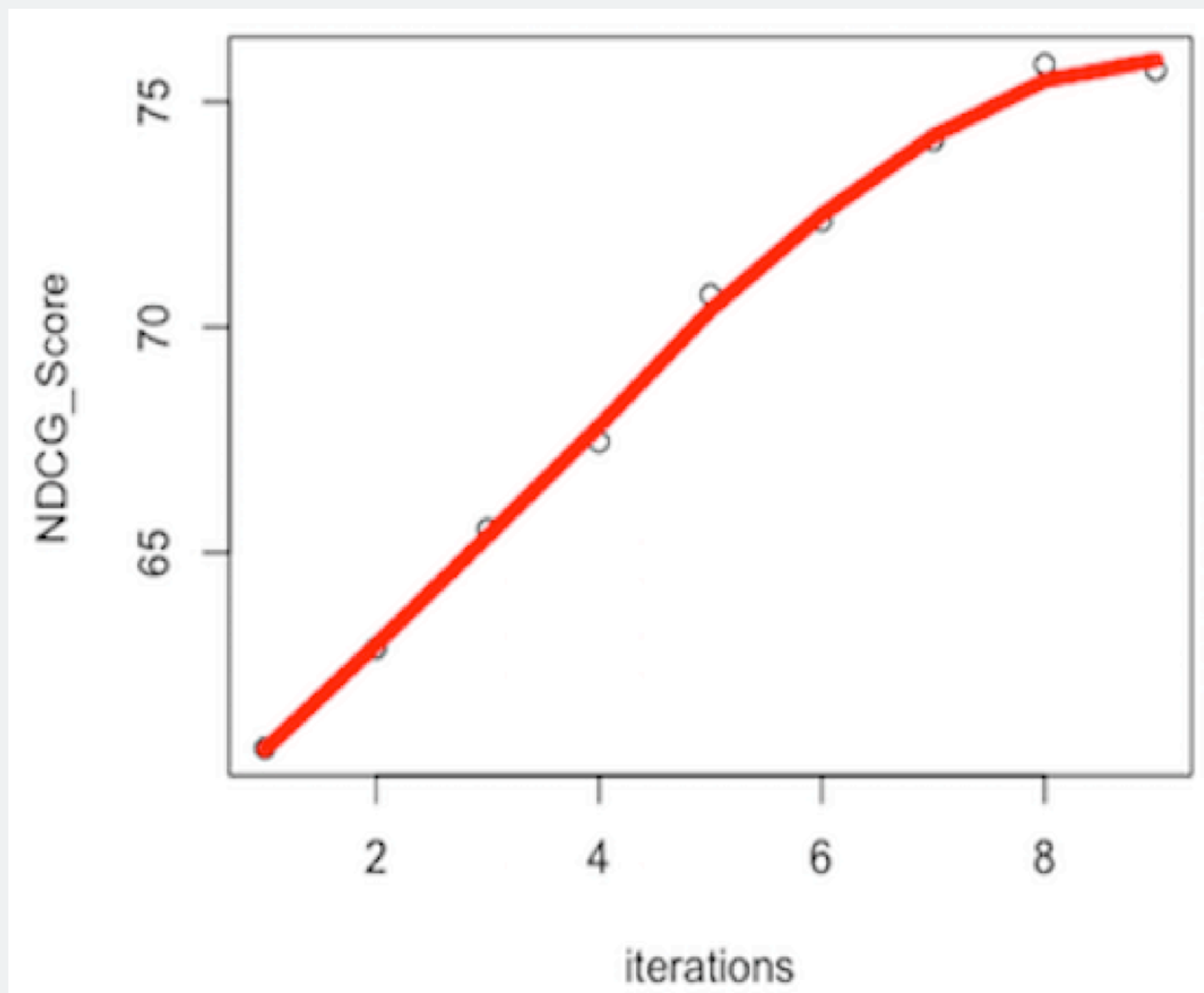
DATASETS AND RESOURCES

- Documentation: https://uhh-lt.github.io/par4sim/
- Datasets: https://uhh-lt.github.io/par4sim/2018/05/29/dataset.html
- Demo: https://ltmaggie.informatik.uni-hamburg.de/par4sim/

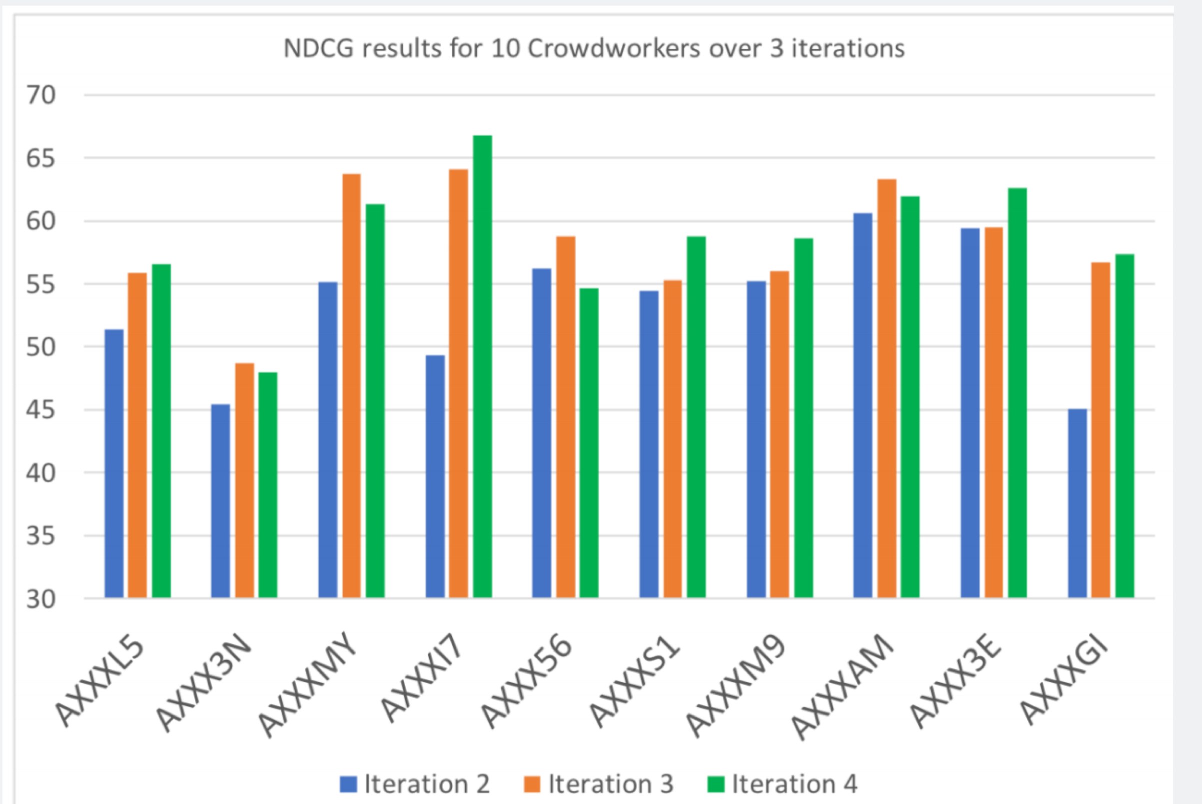
EXPERIMENTAL RESULTS

Testing	NDCG@10								
	Training instances on previous iterations								
	#sentences	baseline	1	≤ 2	≤3	≤4	≤5	≤6	≤7	≤8
1	115	-	-	-	-	-	-	-	-	-
2	214	60.66	62.88	-	-	-	-	-	-	-
3	207	61.05	63.39	65.52	-	-	-	-	-	-
4	210	58.21	60.73	65.93	67.46	-	-	-	-	-
5	233	56.10	62.53	65.66	66.00	70.72	-	-	-	-
6	215	62.18	61.05	66.51	67.86	69.88	72.36	-	-	-
7	213	57.00	62.07	64.02	64.88	67.28	69.27	74.14	-	-
8	195	56.56	59.53	62.11	63.03	64.54	67.40	71.05	75.83	-
9	224	56.14	63.48	65.58	65.87	69.18	69.51	71.31	71.40	75.70

NDCG@10 results for each iteration of the testing instances using training instances from the previous iteration. For example, for testing at iteration 2, the NDCG@10 result using training data from the previous iteration, i.e. iteration 1, is 62.88. The baseline column shows the performance in each iteration using the generic paraphrasing dataset used to train the baseline ranking model.



Learning curve showing the increase of NDCG@10 score over 9 iterations.



NDCG@10 over 3 iterations for 10 workers

DEPARTMENT OF INFORMATICS
LANGUAGE TECHNOLOGY GROUP
http://lt.informatik.uni-hamburg.de/