Visual Indexing of Chronicles

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Anguelos Nicolaou, Vincent Christlein

Outline

- Introduction
- System overview
- Objectives
- Outlook

Introduction

Who are we?

Anguelos Nicolaou
Post-Doc Researcher at **Pattern Recognition Lab**

Research Areas:
- Computer Vision
- Image Texture Analysis
- Document Analysis

Email: anguelos.nicolaou@gmail.com

Dr.-Ing. Vincent Christlein
Post-Doc Researcher at **Pattern Recognition Lab**

Research Areas:
- Computer Vision
- Computational Humanities
- Document Analysis

Web: *https://lme.tf.fau.de/person/christlein/*
Email: vincent.christlein@fau.de

Where are we?

Who are you?

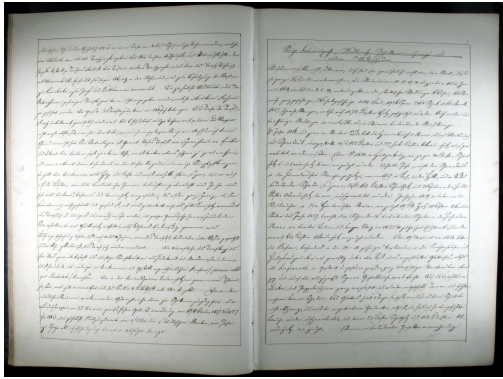
- Scholars/Users:
  - System Limitations?
  - System Potential?
  - Feature Requests?
  - Explaining Problems?
- Managers / Decision Makers:
  - What is easy?
  - What is hard?
  - Who can do what?
- Technical people:
  - How to integrate?
  - Resource sharing?
  - Code-basis Sharing?

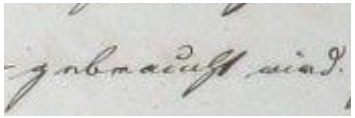
Who are you?

- Scholars/Users:
  - System Limitations?
  - System Potential?
  - Feature Requests?
  - Explaining Problems?
- Managers / Decision Makers:
  - What is easy?
  - What is hard?
  - Who can do what?
- Technical people:
  - How to integrate?
  - Resource sharing?
  - Code-basis Sharing?

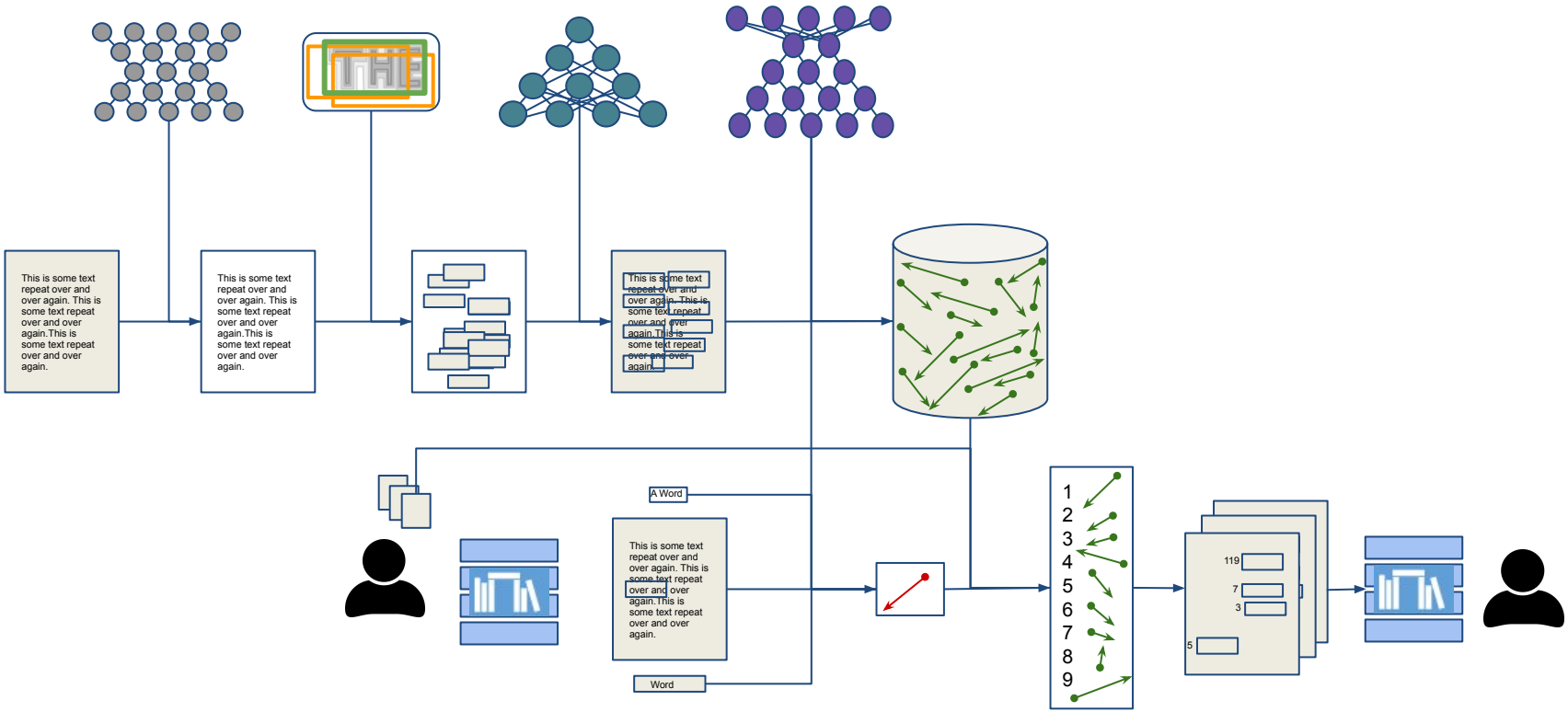
Motivation:

- Can we read manuscripts? Not yet!
- So what can we do without reading?
  - Retrieval: we will show all possible answers to the user but in the **best order** possible
- Constraints:
  - Robust+modular design
  - User time
  - Big data: 6620 Chronicles → 769052 pages → 1B embeddings
  - Harvesting ambiguity: Yes/No → Probably/Maybe
  - Out of dictionary words
- Caching tradeoffs: Do I save something or recompute

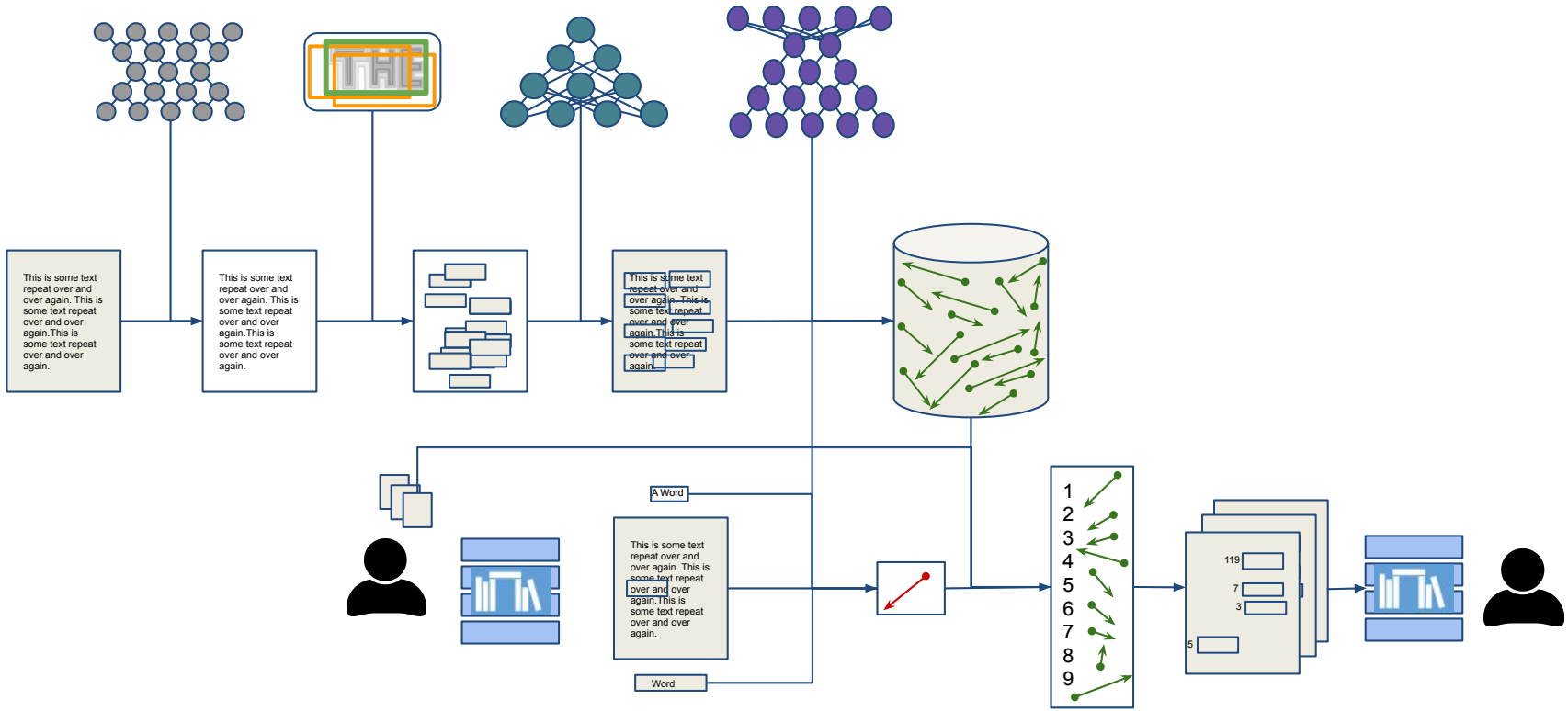




Pipeline Overview

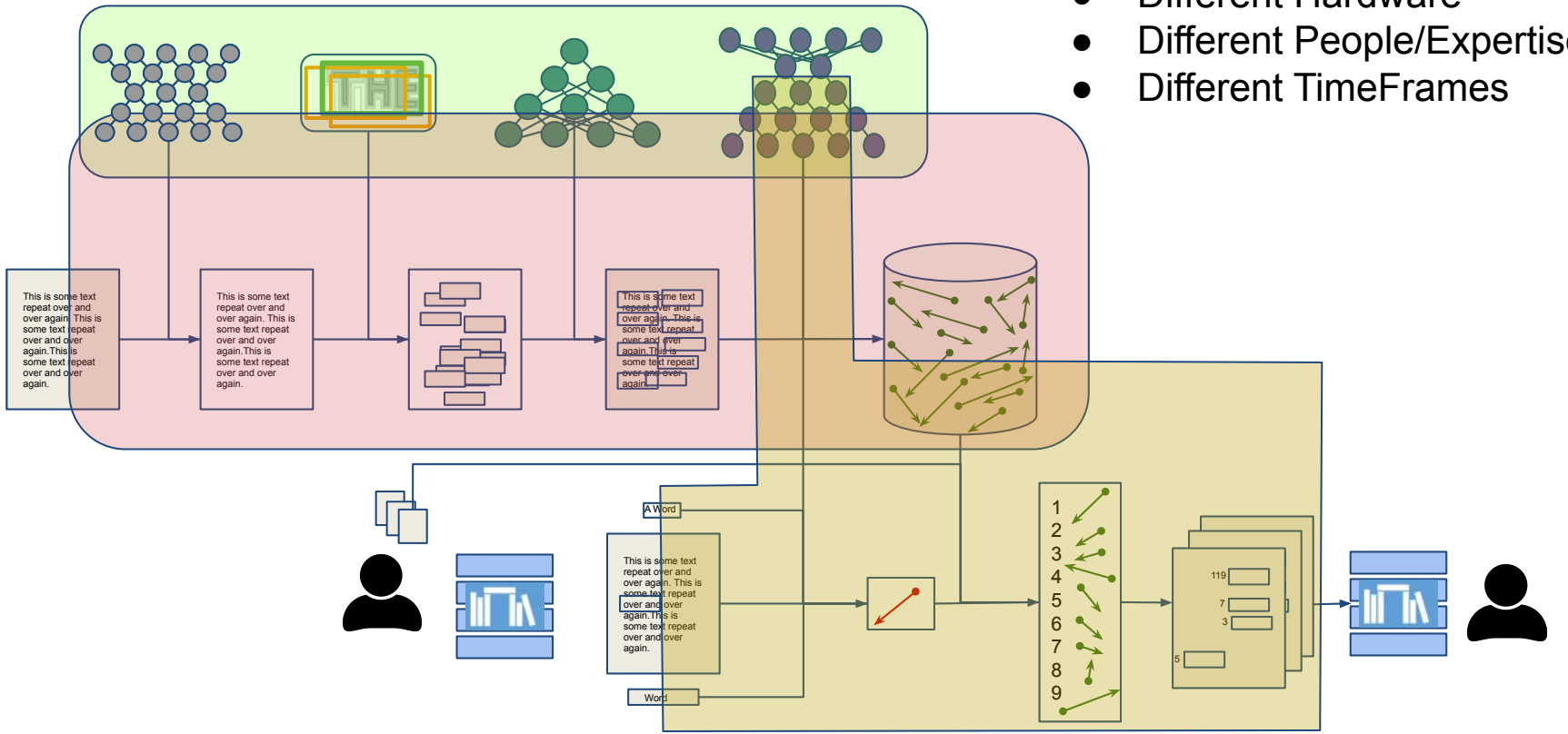


Generic Slide

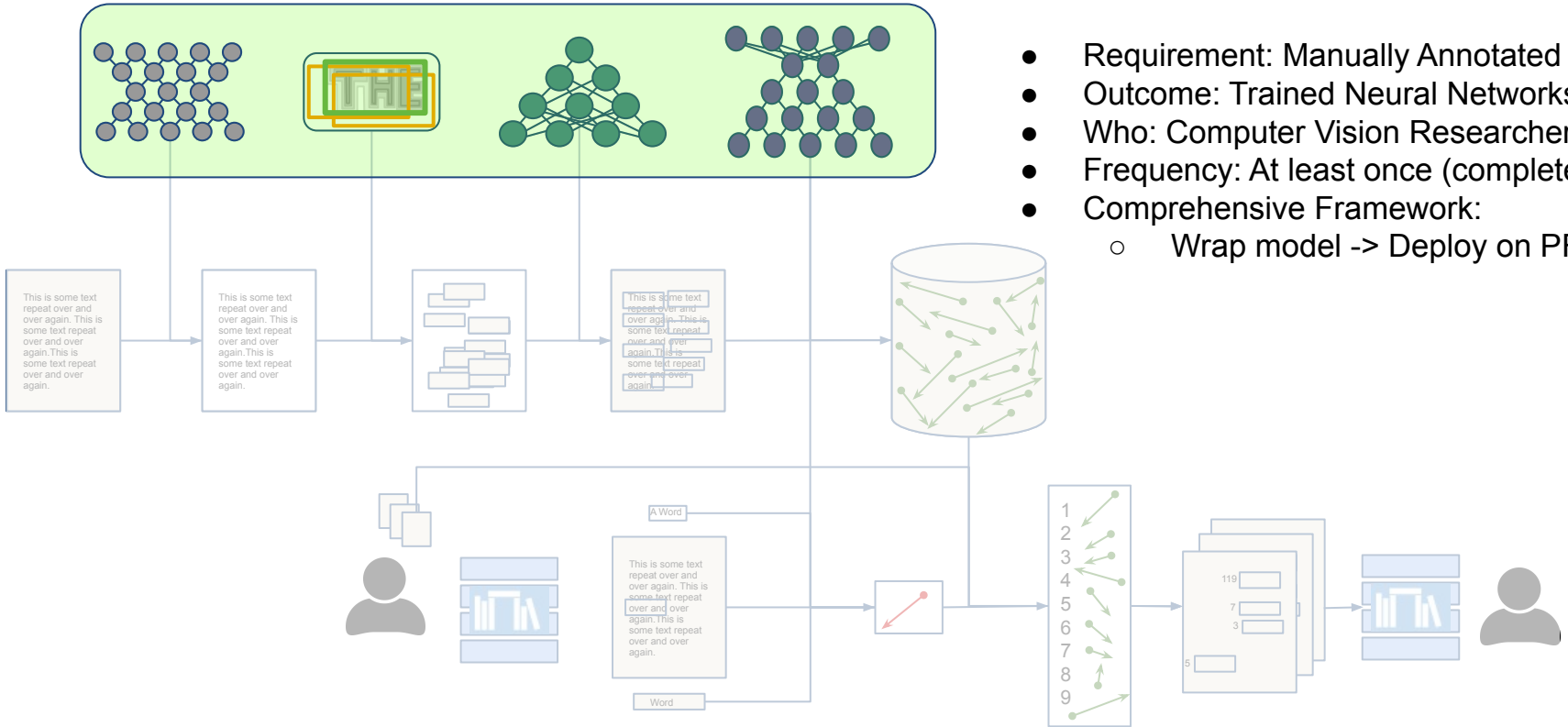


Subsystems

- Totally Independent
- Different Hardware
- Different People/Expertise
- Different TimeFrames

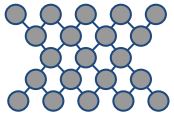


Subsystems: Machine Learning

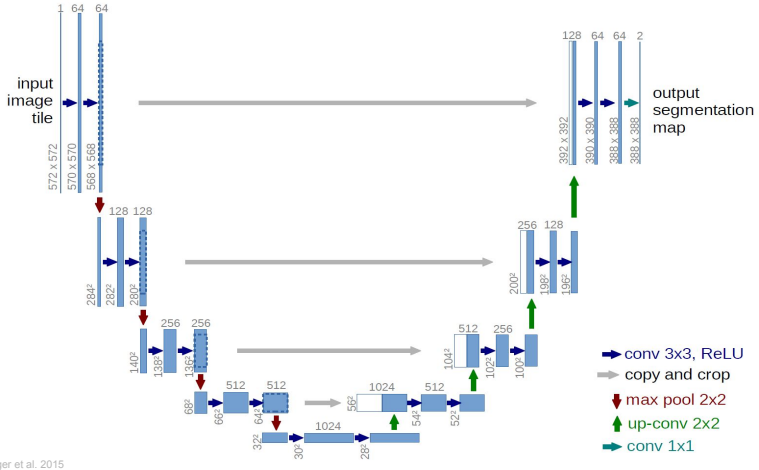


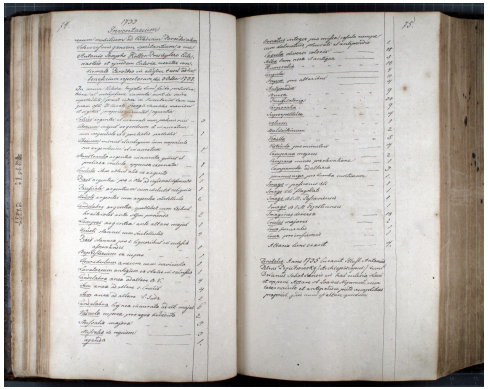
- Requirement: Manually Annotated Data
- Outcome: Trained Neural Networks
- Who: Computer Vision Researcher
- Frequency: At least once (completed)
- Comprehensive Framework:
  - Wrap model -> Deploy on PF

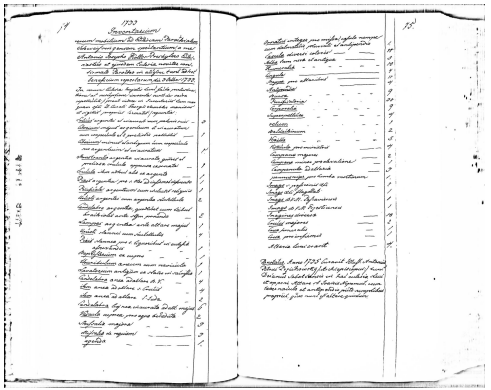
Machine Learning: Segmentation



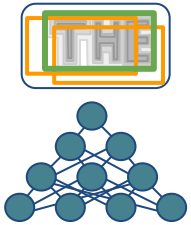
- UNet architecture
- Trained on external data
- Probabilistic outputs



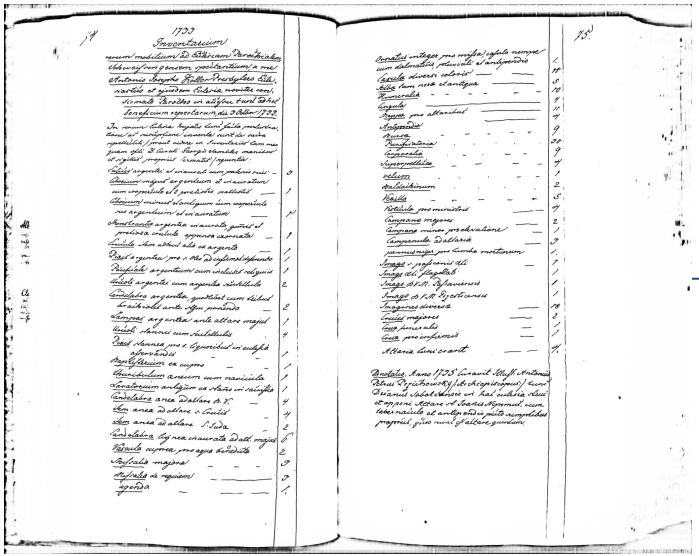


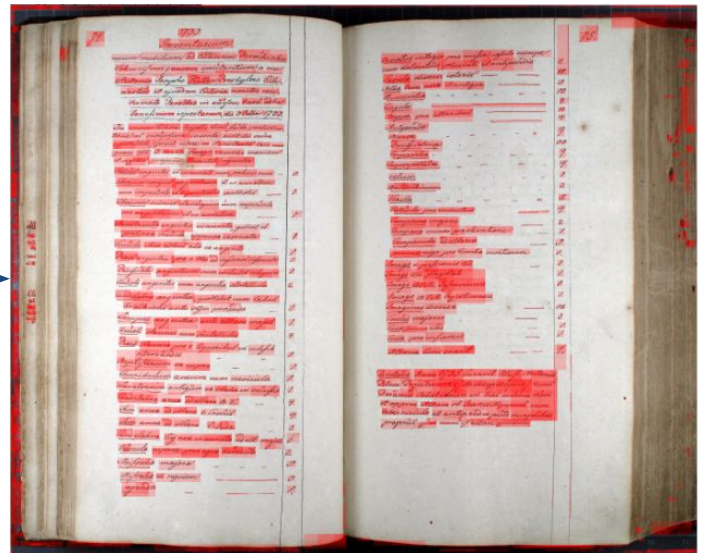


Machine Learning: Word Proposals

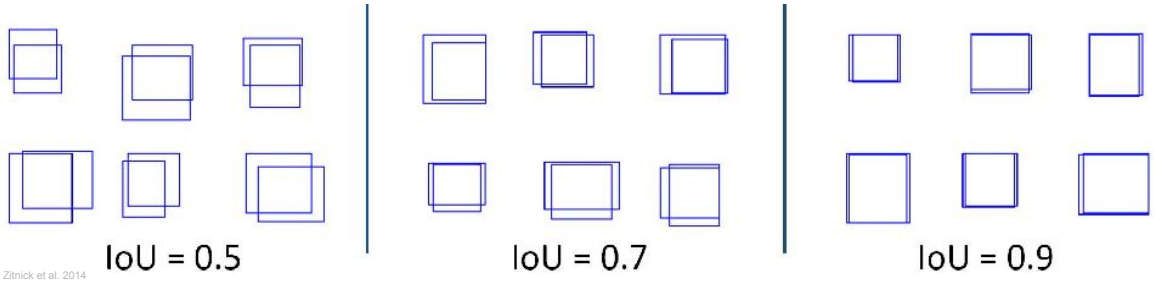


- Connected Component Analysis
- Deep Multilayer Perceptron on box properties
- Storage/RAM/Computation vs. Performance



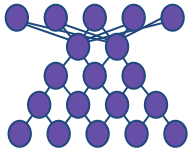


Word Proposals Performance

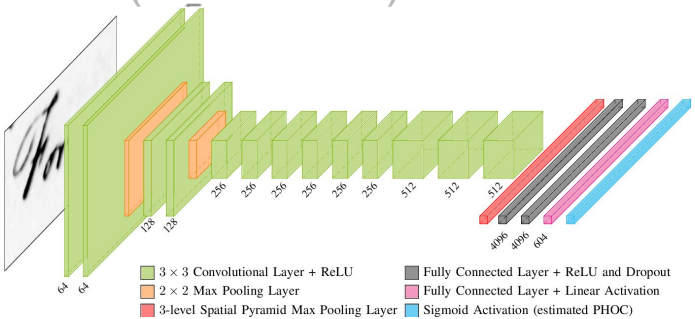


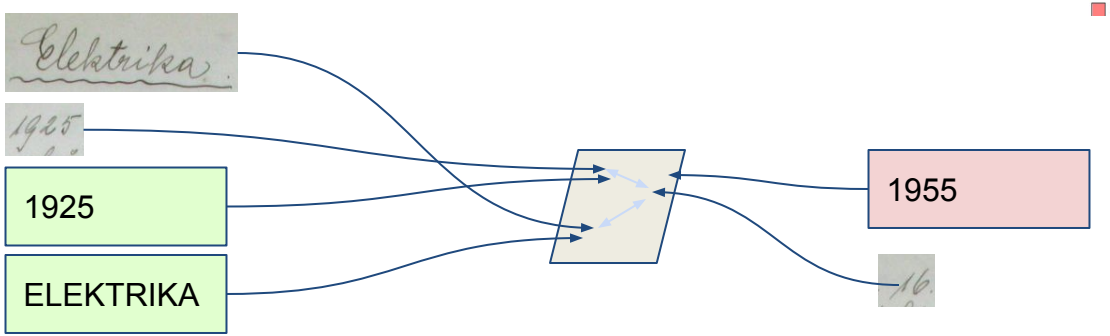
Corpus	Existing	Proposed	Recall @ 0.9	Recall @ 0.75	Recall @ 0.5
Chudenice_2	2520	12965	1.74%	42.46%	**82.82%**
Plasy	6397	35468	0.07%	16.48%	**83.77%**
Svojsin_2	4328	21102	0.39 %	32.97%	**82.67%**

Machine Learning: Word Embeddings

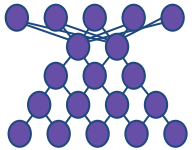


- Deep Residual Network
- Embedding: Word Image -> 540 numbers (dimensions)
- 540 -> 2KB -> 500MB
  - RAM
  - CPU
  - Storage
- PHOC: Word -> 540 numbers

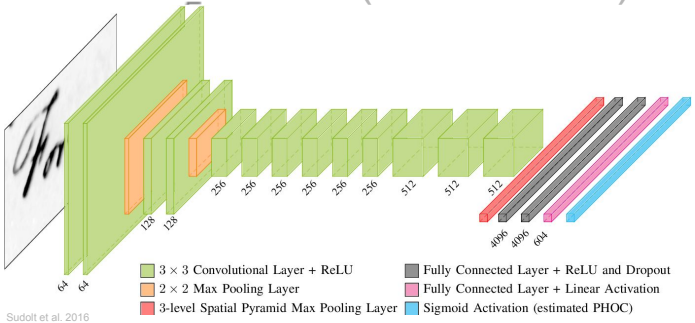




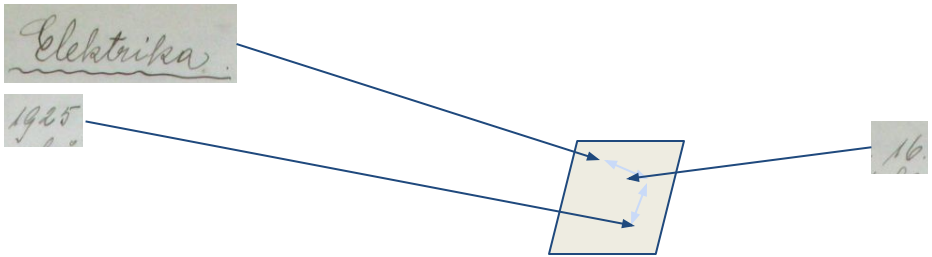
Machine Learning: Word Embeddings



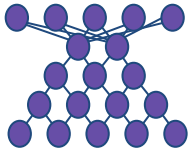
- Deep Residual Network
- Embedding: Word (proposal) image → 540 numbers (dimensions)
- 540 → 2KB → 500MB
  - RAM
  - CPU
  - Storage



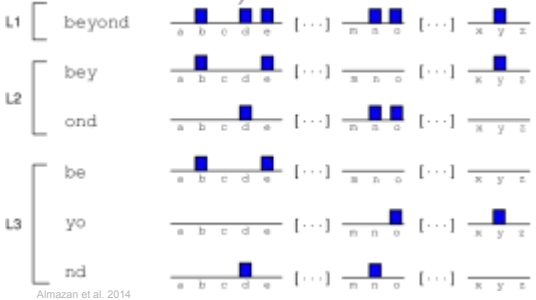
Sudolt et al. 2016



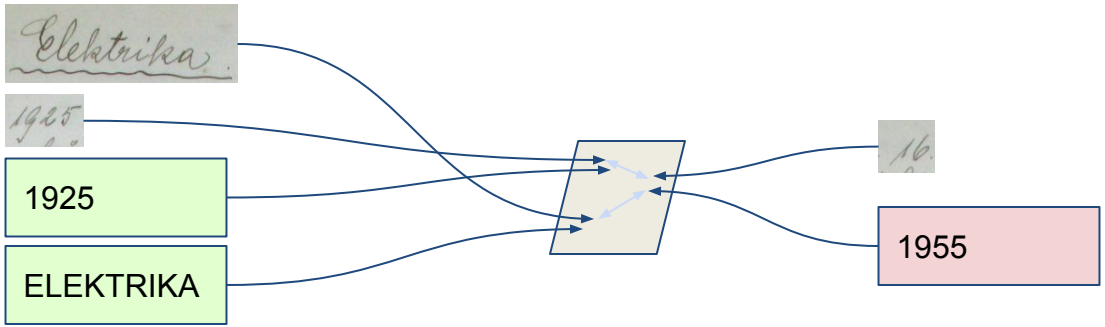
Machine Learning: Word Embeddings



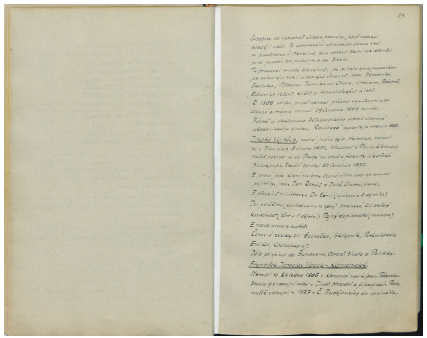
- Deep Residual Network
- Embedding: Word Image → 540 numbers (dimensions)
- 540 → 2KB → 500MB
  - RAM
  - CPU
  - Storage
- PHOC: Word → 540 numbers

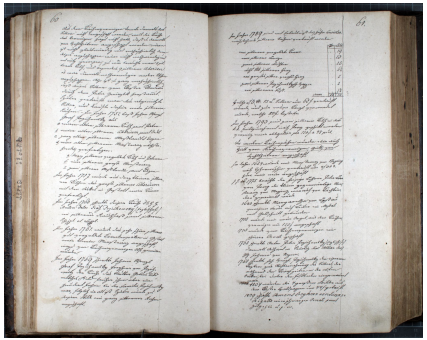


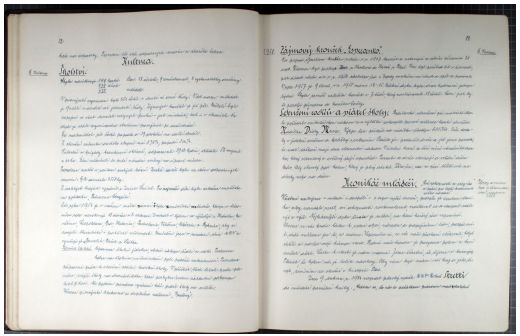
Almazan et al. 2014



Word Embedding Performance



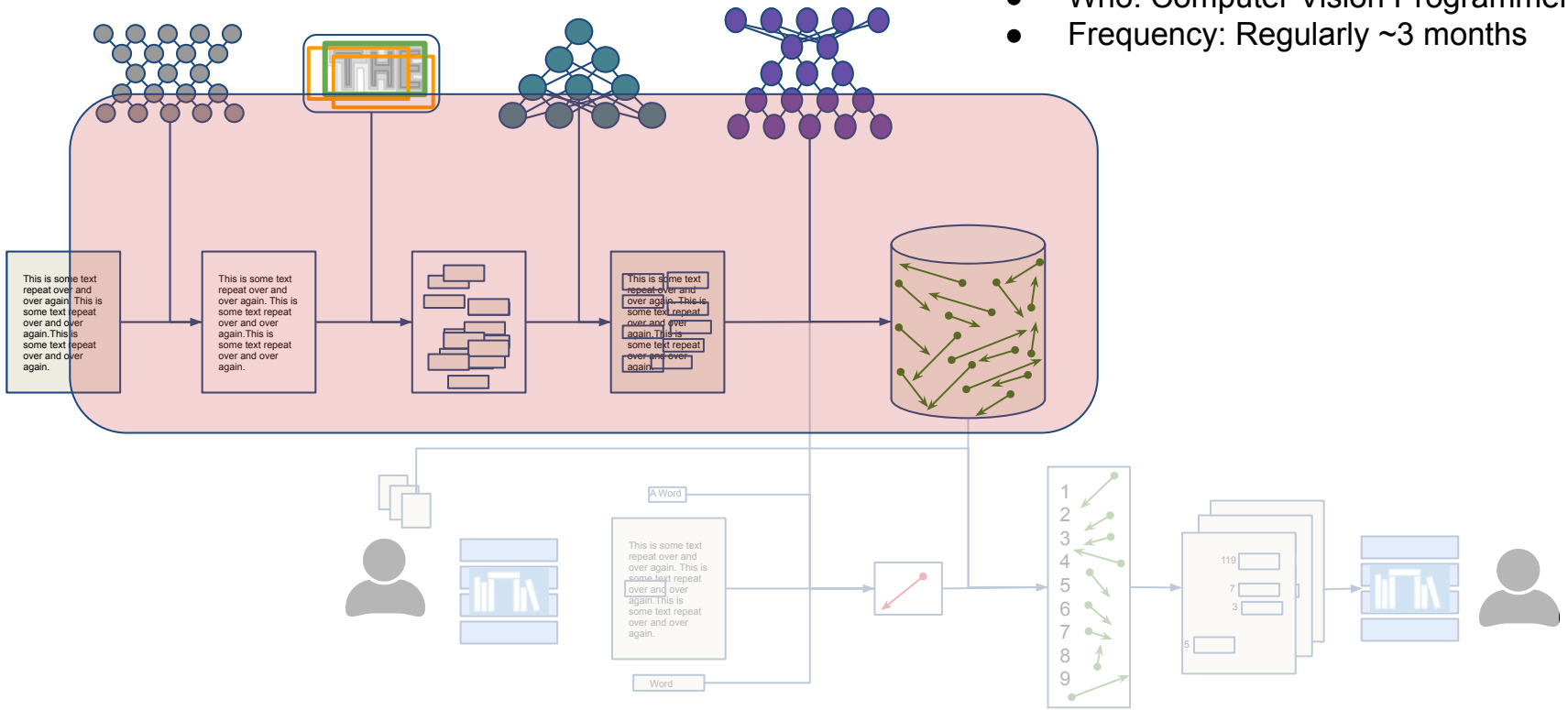




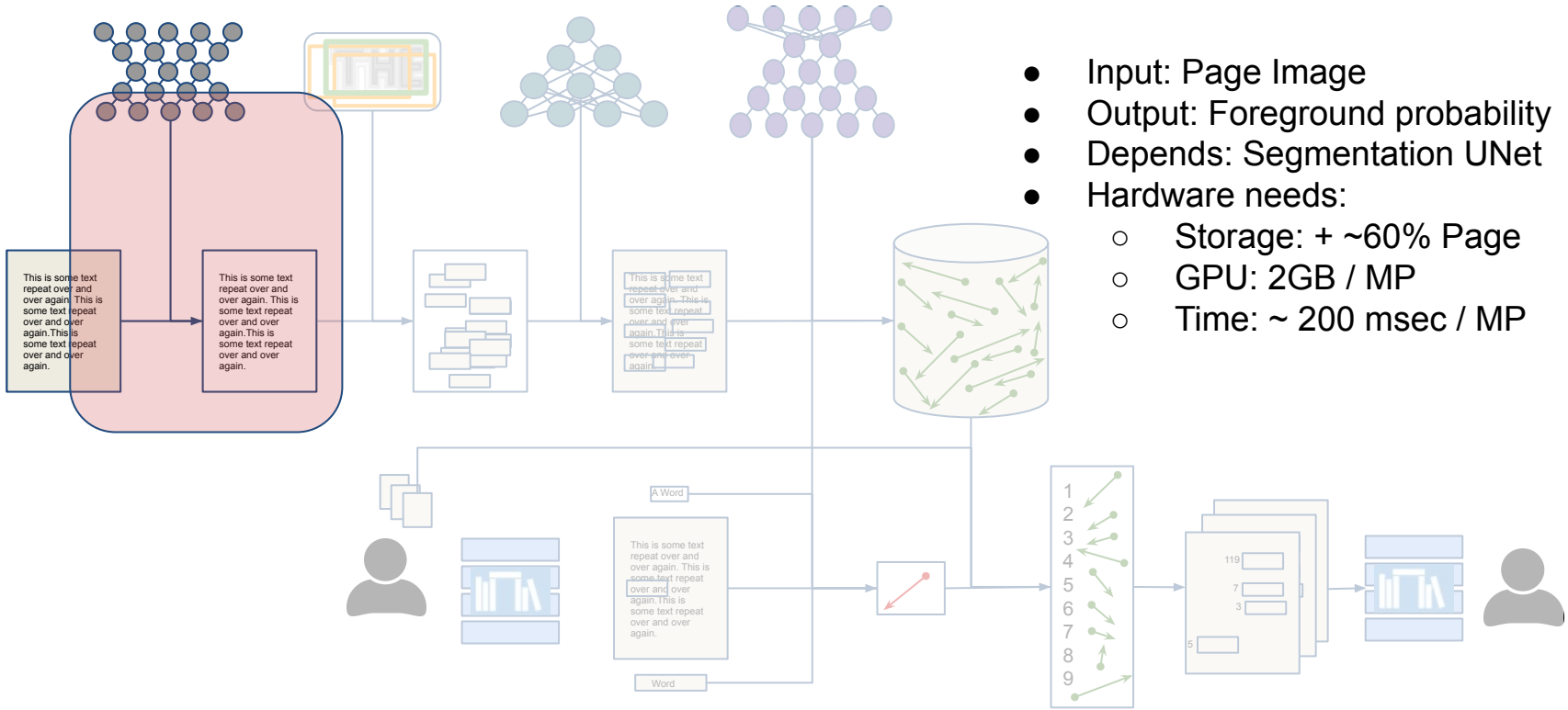
Chronicle	Language	Words	Recall @ 10	Recall @ 100	Accuracy	mAP
Chudenice_2	Czech	1408	81.503%	83.90%	76.35%	77.70%
Plasy	Czech	3486	44.47%	63.65%	25.33%	30.22%
Svojsin_2	German	1558	54.78%	58.46%	51.80%	52.83%

Subsystems: Offline

- Requirement: Trained Neural Networks
- Outcome: Visual Word Index (Database)
- Who: Computer Vision Programmer
- Frequency: Regularly ~3 months

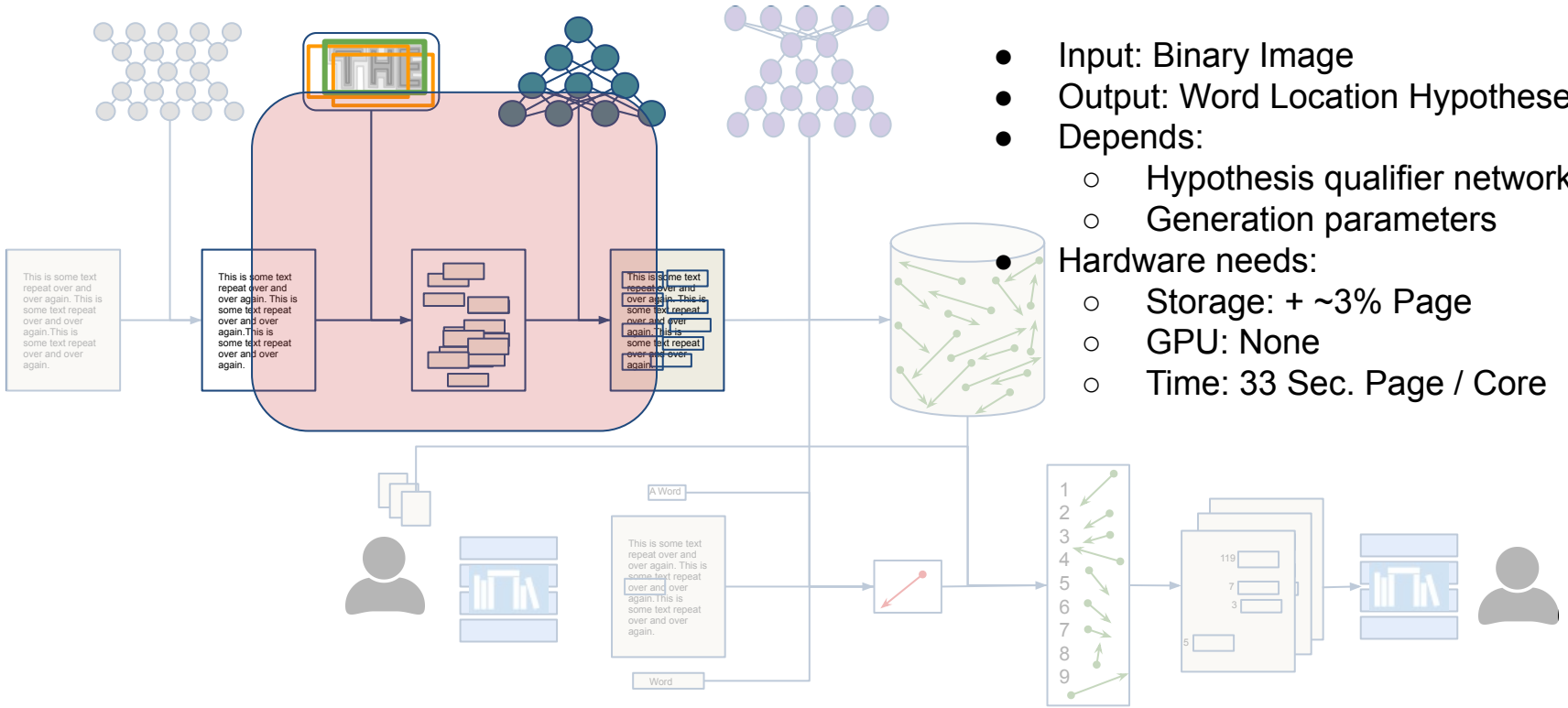


Offline Subsystem: Segmentation



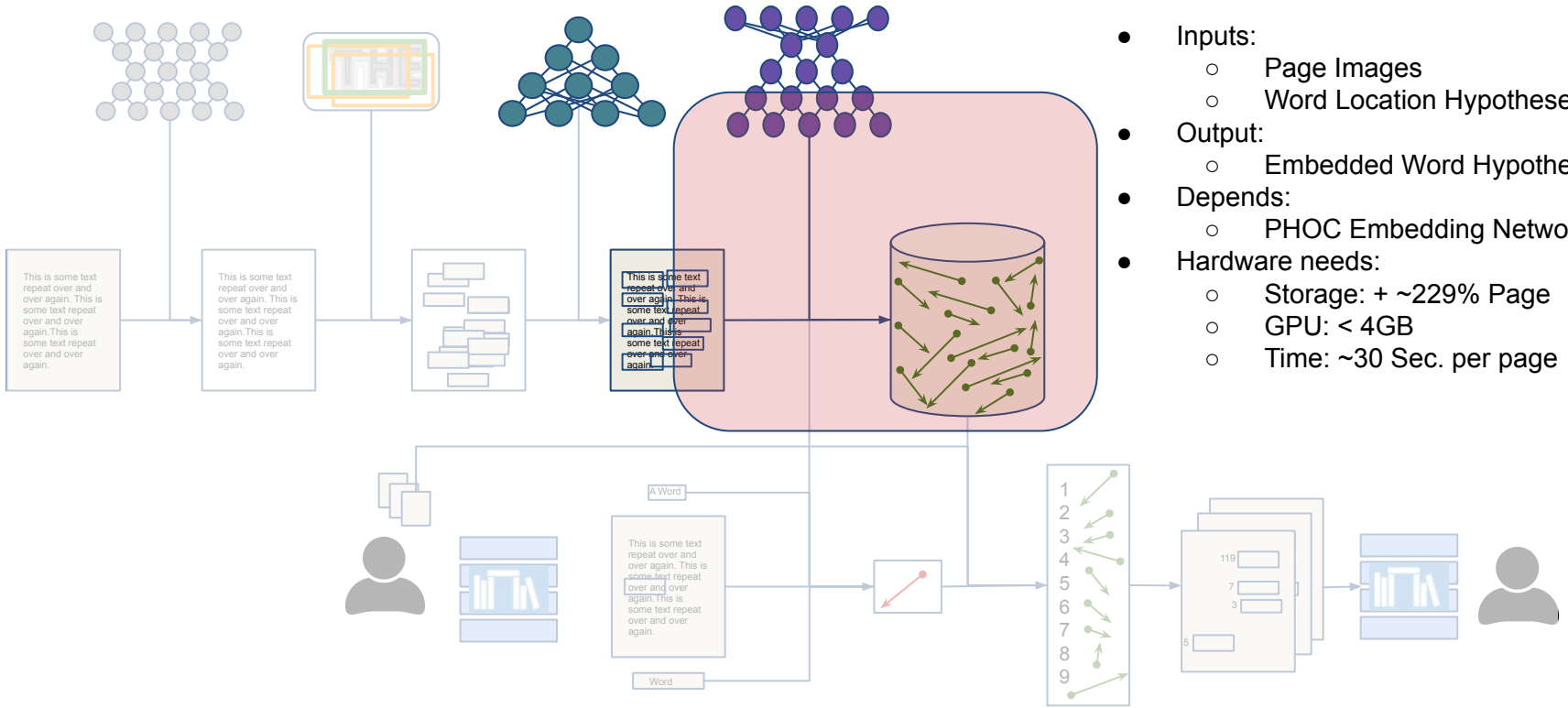
- Input: Page Image
- Output: Foreground probability
- Depends: Segmentation UNet
- Hardware needs:
  - Storage: + ~60% Page
  - GPU: 2GB / MP
  - Time: ~ 200 msec / MP

Offline Subsystems: Word proposals



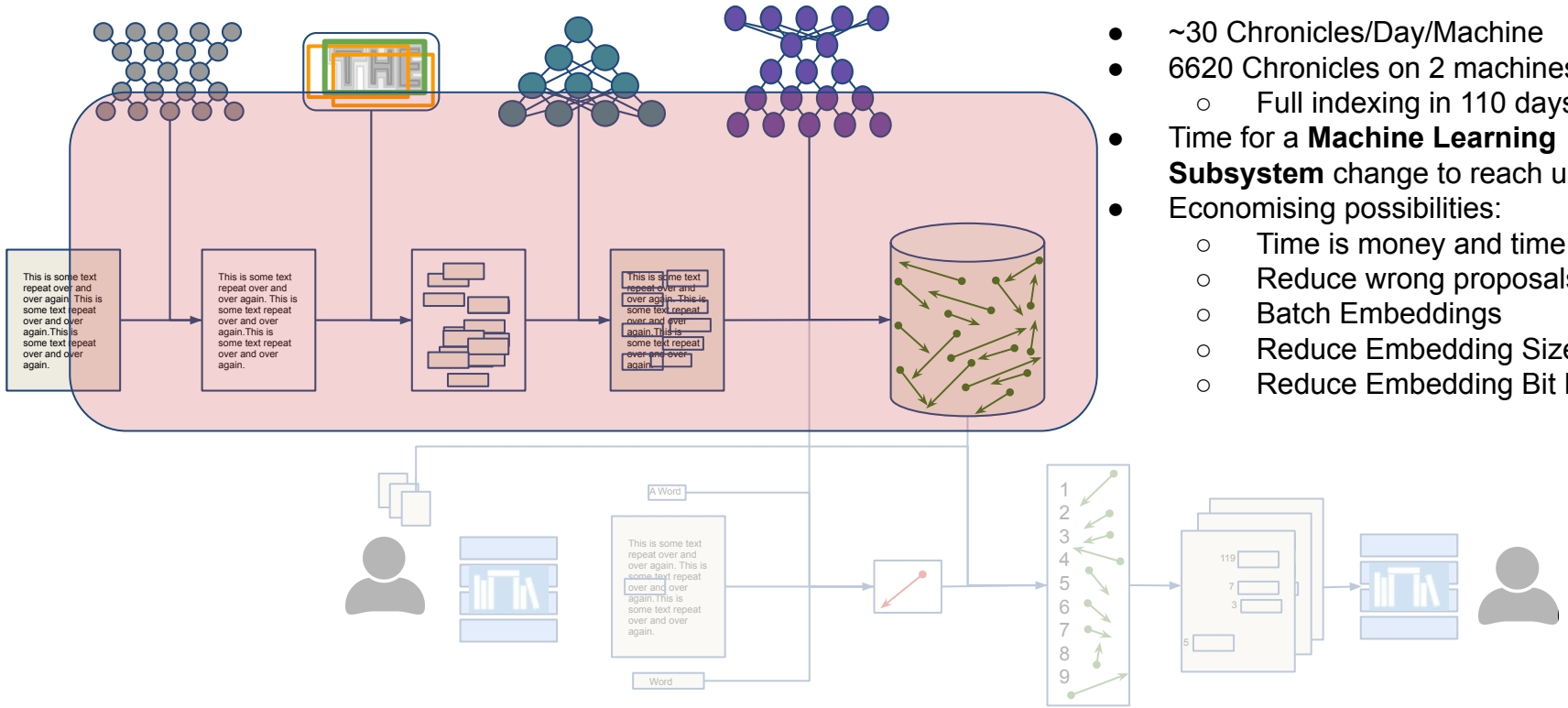
- Input: Binary Image
- Output: Word Location Hypotheses
- Depends:
  - Hypothesis qualifier network
  - Generation parameters
- Hardware needs:
  - Storage: + ~3% Page
  - GPU: None
  - Time: 33 Sec. Page / Core

Offline Subsystems: Word Embeddings



- Inputs:
  - Page Images
  - Word Location Hypotheses
- Output:
  - Embedded Word Hypotheses
- Depends:
  - PHOC Embedding Network
- Hardware needs:
  - Storage: + ~229% Page
  - GPU: < 4GB
  - Time: ~30 Sec. per page

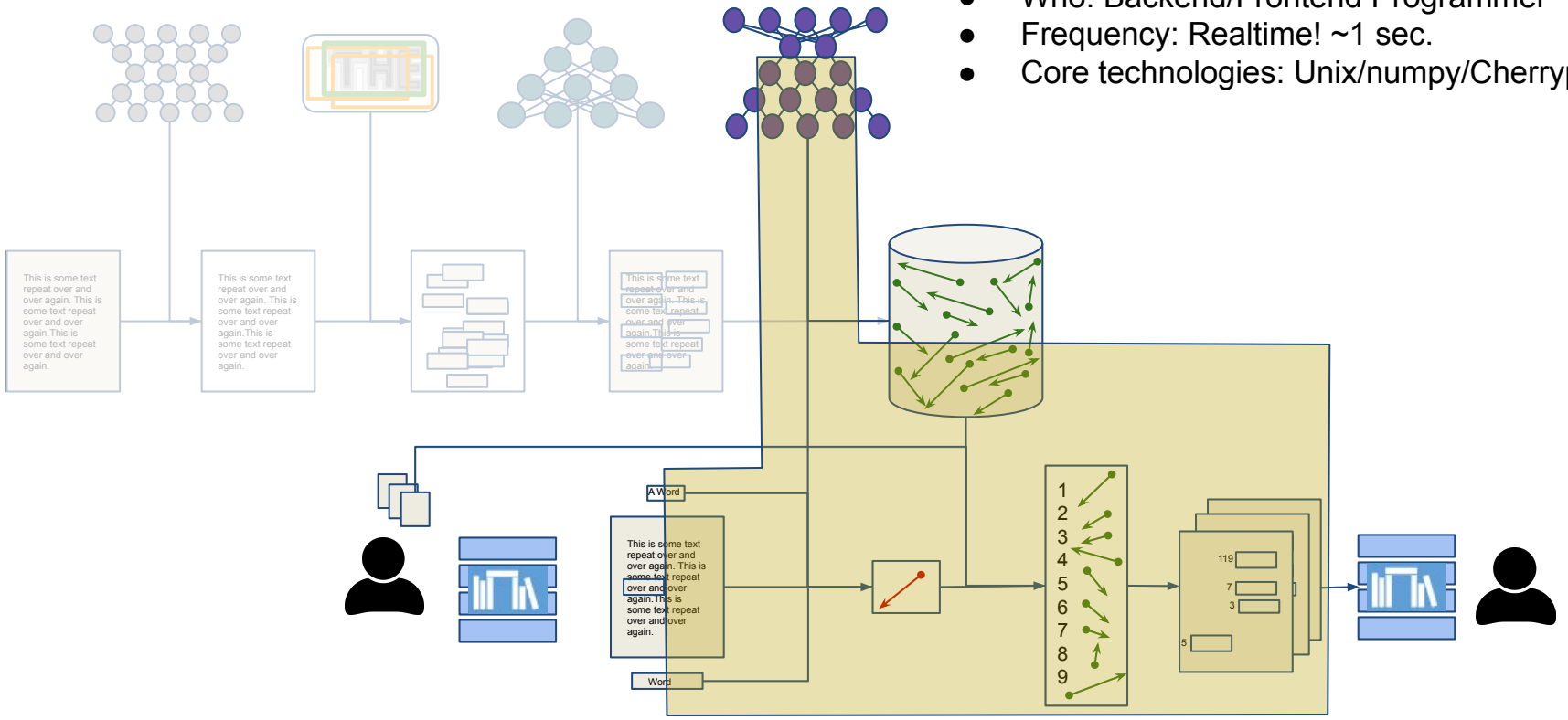
Offline Subsystem: Throughput



- ~30 Chronicles/Day/Machine
- 6620 Chronicles on 2 machines?
  - Full indexing in 110 days.
- Time for a **Machine Learning Subsystem** change to reach users.
- Economising possibilities:
  - Time is money and time
  - Reduce wrong proposals
  - Batch Embeddings
  - Reduce Embedding Sizes
  - Reduce Embedding Bit Depth

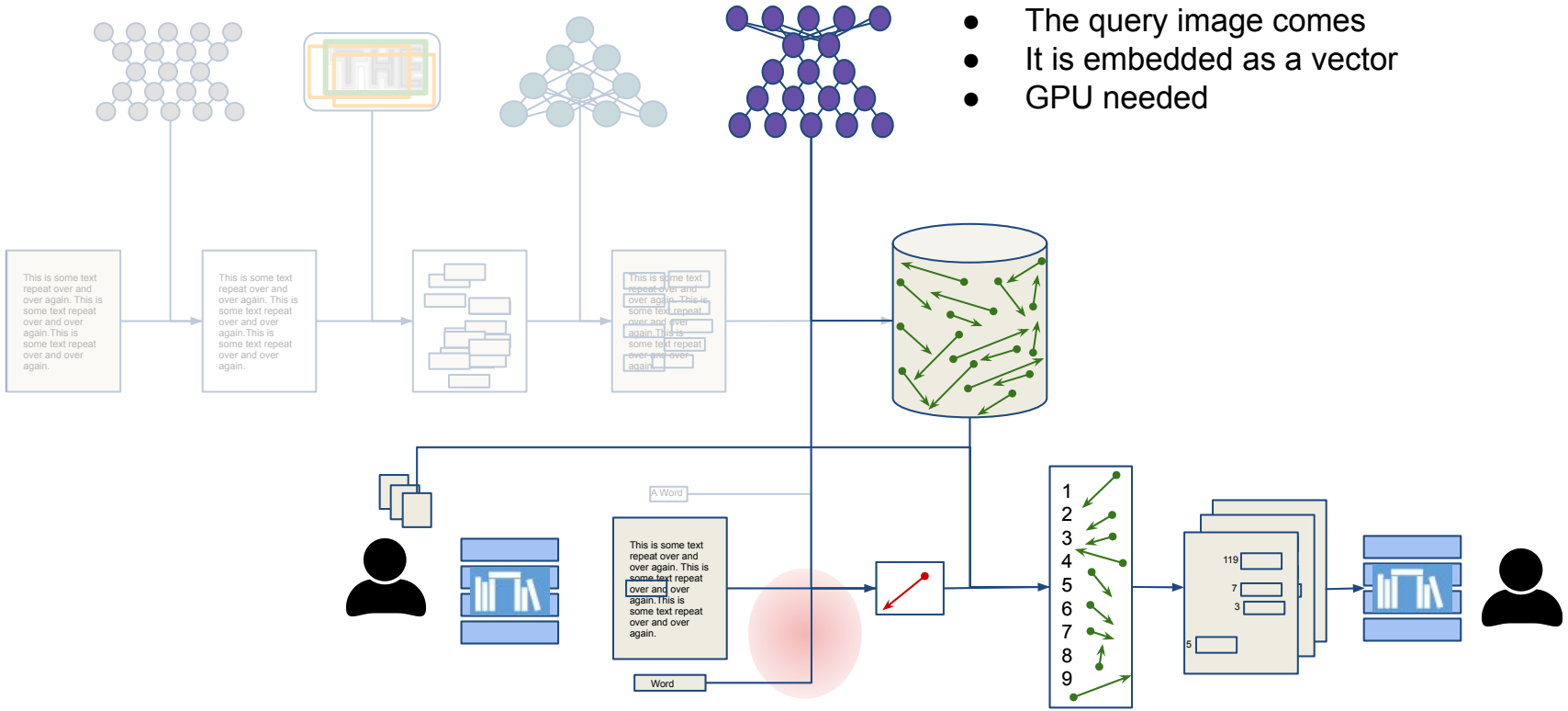
Subsystems: Online

- Requirement: Visual Word Index (Database)
- Outcome: Answers To User Queries
- Who: Backend/Frontend Programmer
- Frequency: Realtime! ~1 sec.
- Core technologies: Unix/numpy/Cherrypy



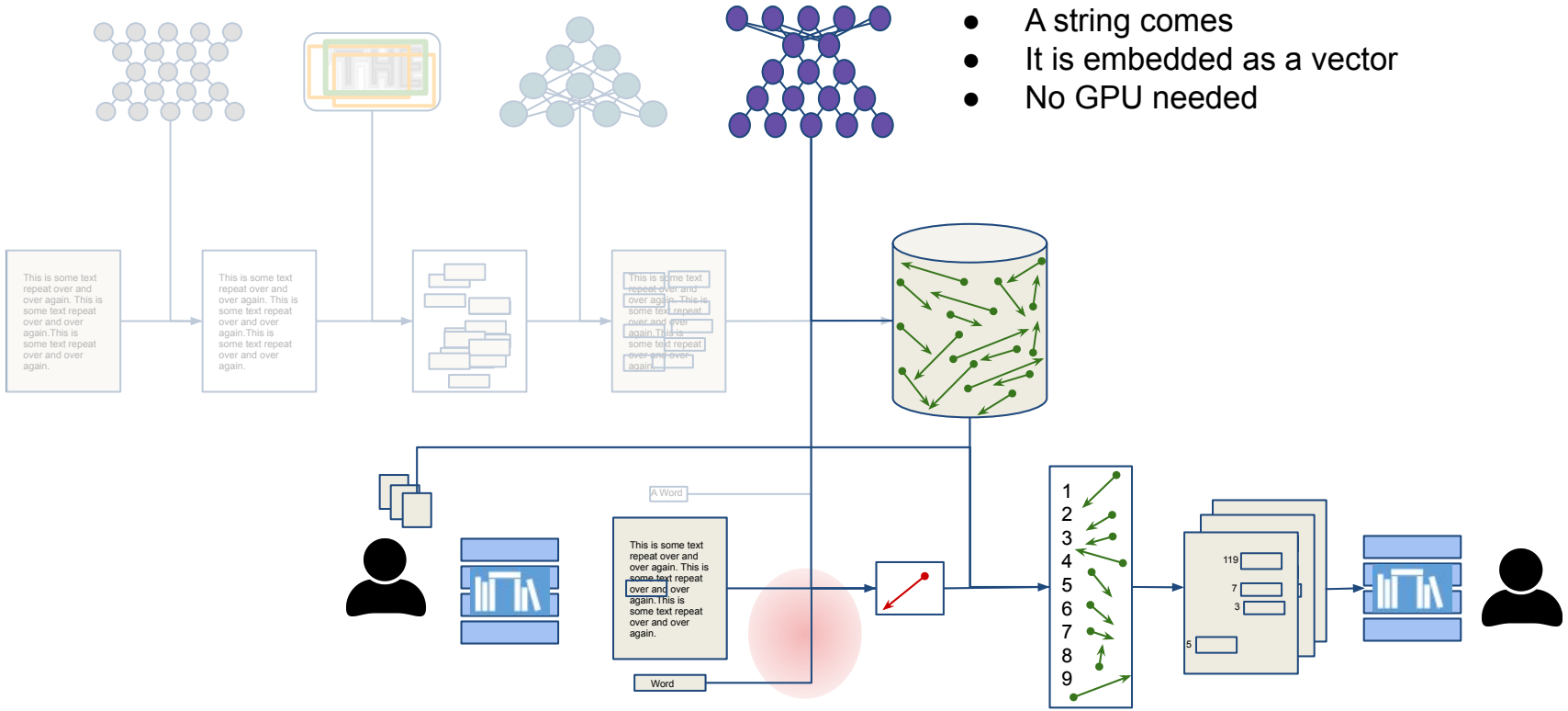
Online Subsystem: Query by Example (QbE)

- The query image comes
- It is embedded as a vector
- GPU needed

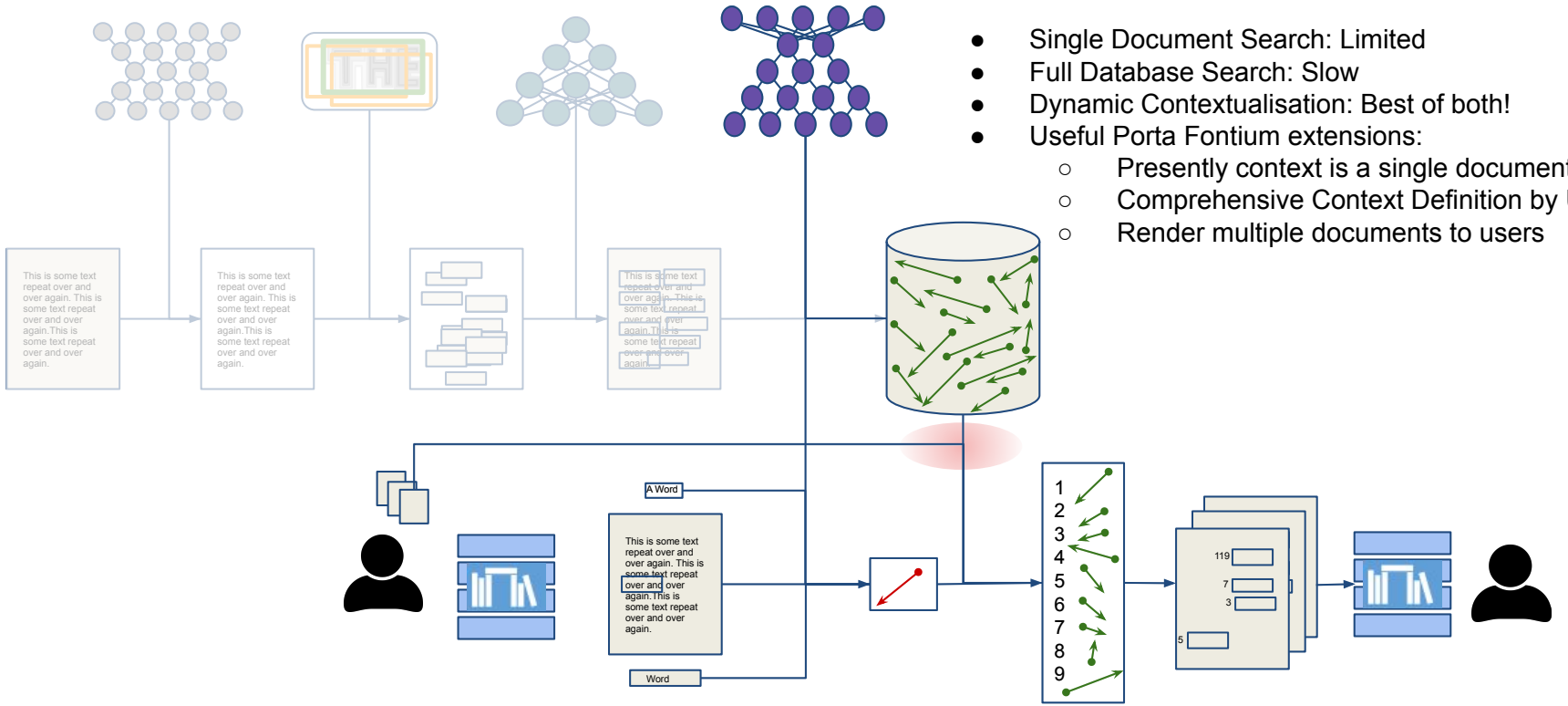


Online Subsystem: Query by String (QbS)

- A string comes
- It is embedded as a vector
- No GPU needed

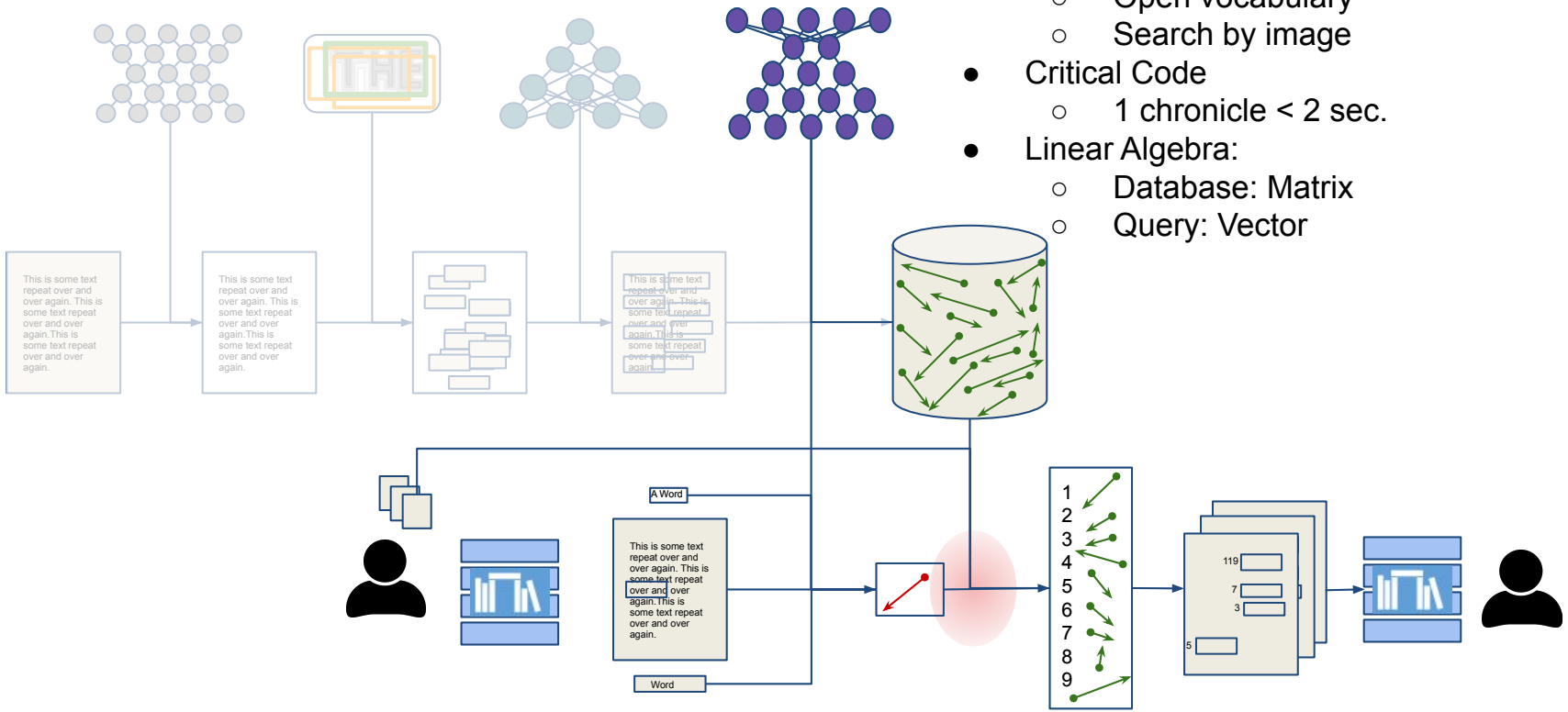


Online Subsystem: Dynamic Contextualisation



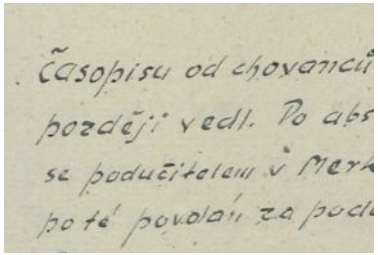
- Single Document Search: Limited
- Full Database Search: Slow
- Dynamic Contextualisation: Best of both!
- Useful Porta Fontium extensions:
  - Presently context is a single document
  - Comprehensive Context Definition by Users
  - Render multiple documents to users

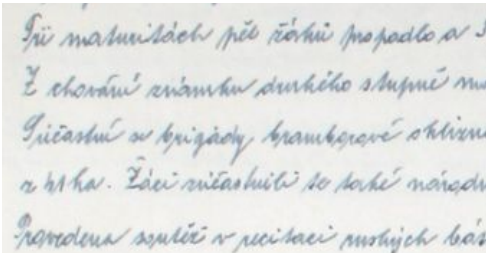
Online Subsystem: Search

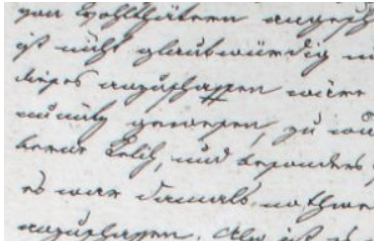


- Why not precompute everything?
  - A custom search for every question
  - Open vocabulary
  - Search by image
- Critical Code
  - 1 chronicle < 2 sec.
- Linear Algebra:
  - Database: Matrix
  - Query: Vector

End-to-end performance Evaluation



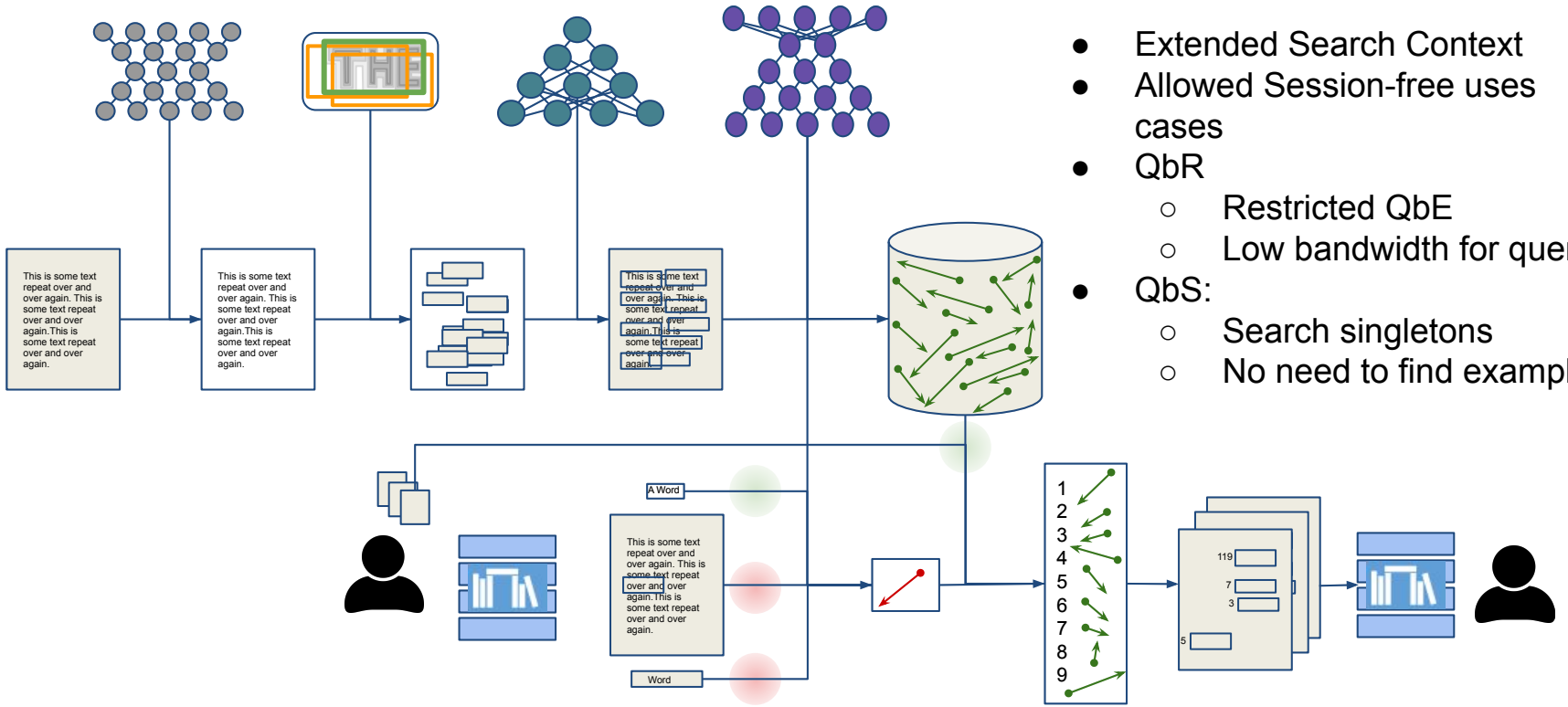




Chronicle	Language	Words	Recall @ 10	Recall @ 100	Accuracy	mAP
Chudenice_2	Czech	1408	34.45%	44.40%	57.53%	30.20%
Plasy	Czech	3486	7.9%	15.12%	14.6%	5.40%
Svojsin_2	German	1558	24.50%	33.38%	38.76%	21.80%

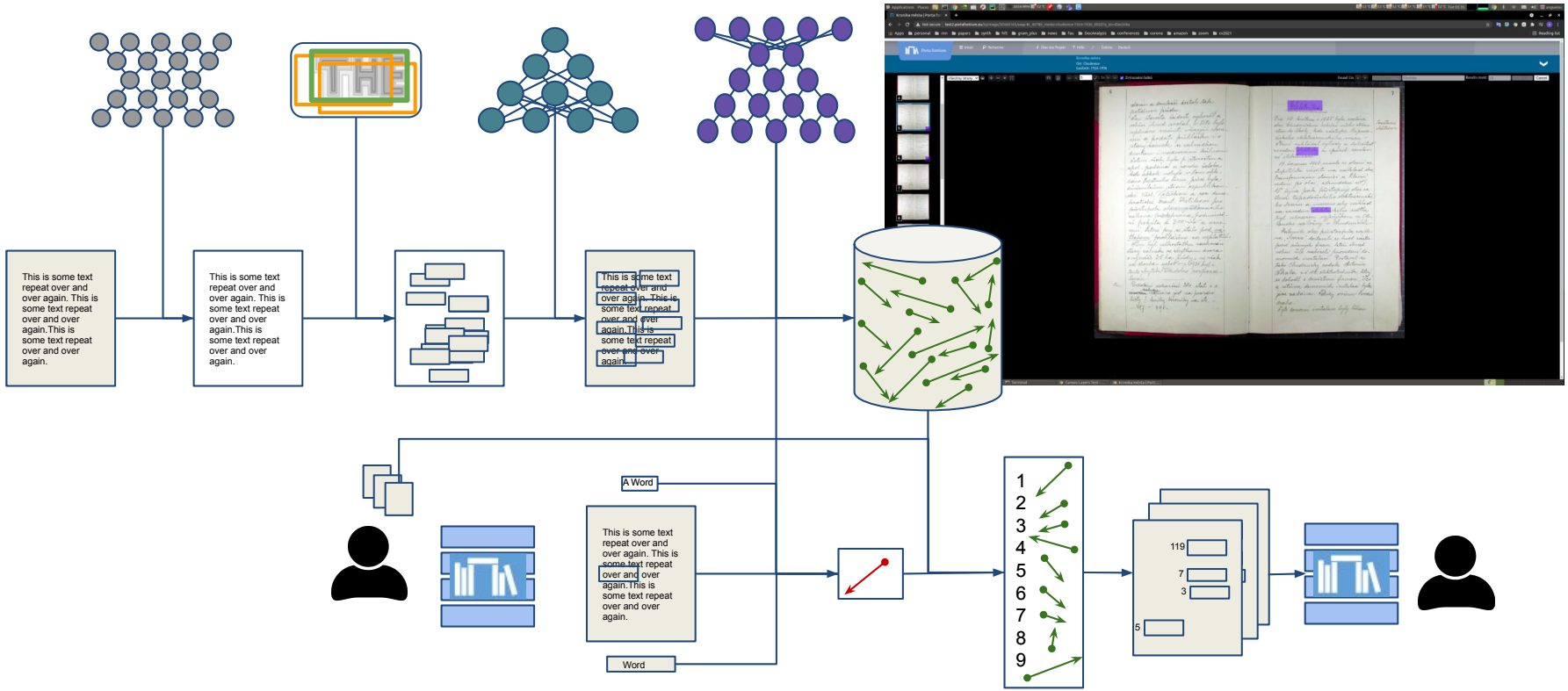
Objectives + Summary

Word Spotting QbE

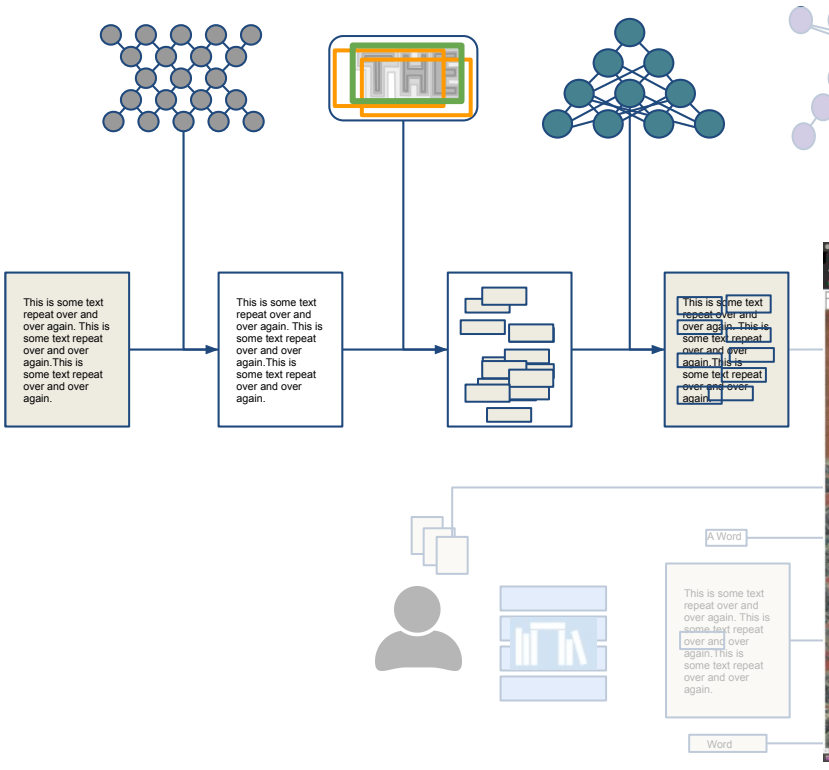


- Extended Search Context
- Allowed Session-free uses cases
- QbR
  - Restricted QbE
  - Low bandwidth for query
- QbS:
  - Search singletons
  - No need to find example

Integration with Porta Fontium



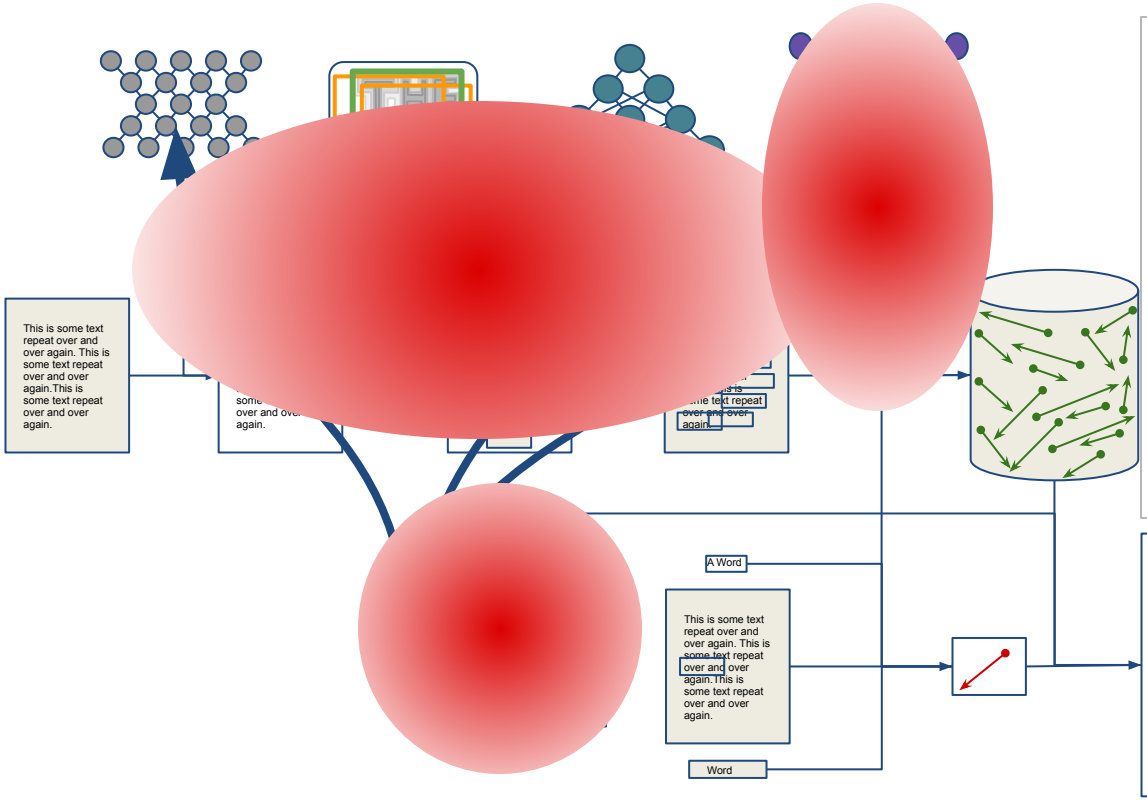
Segmentation Indexing / Text Annotation



- Annotation Tools
- Web Based!
  - Can we integrate to Porta Fontium?
- User's time and knowledge: the most valuable resource

Outlook

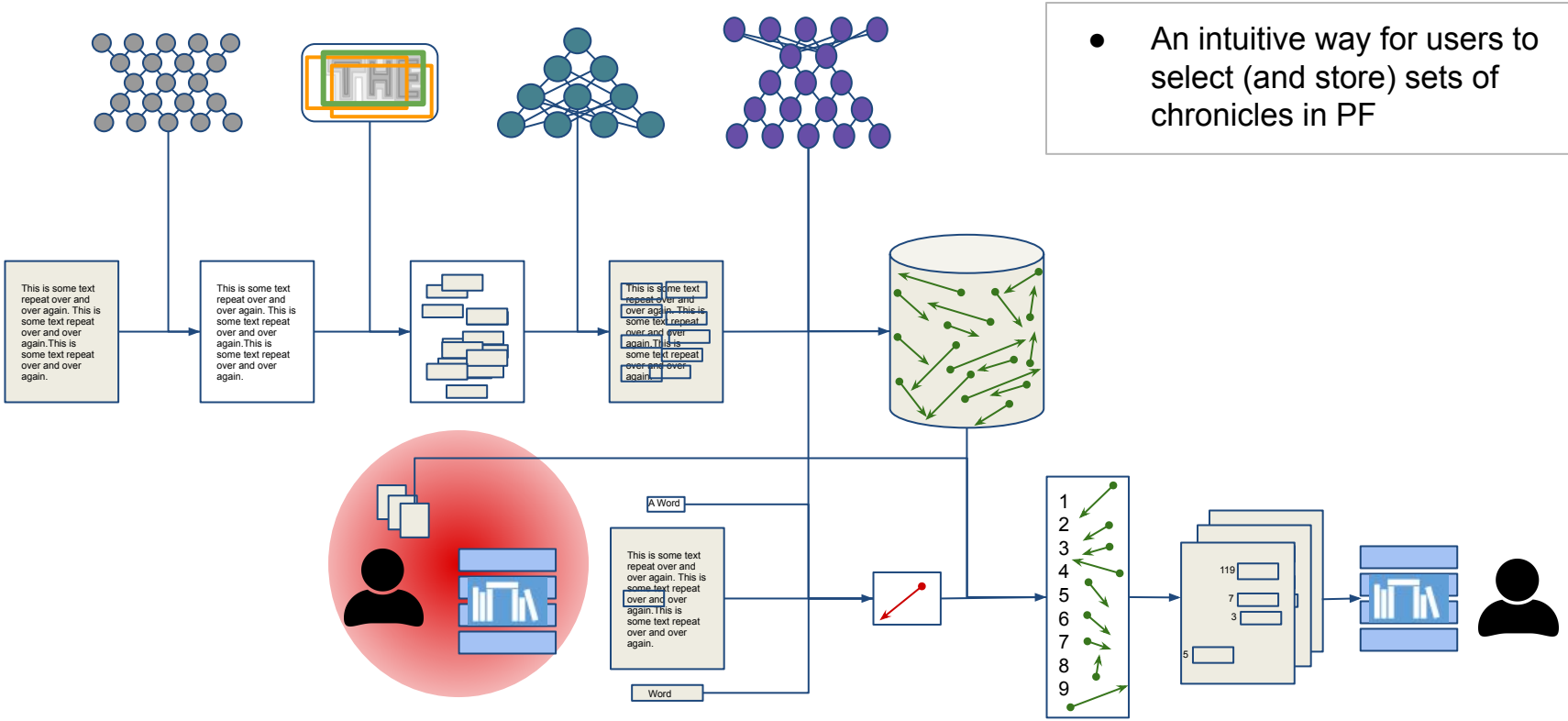
What can be next?



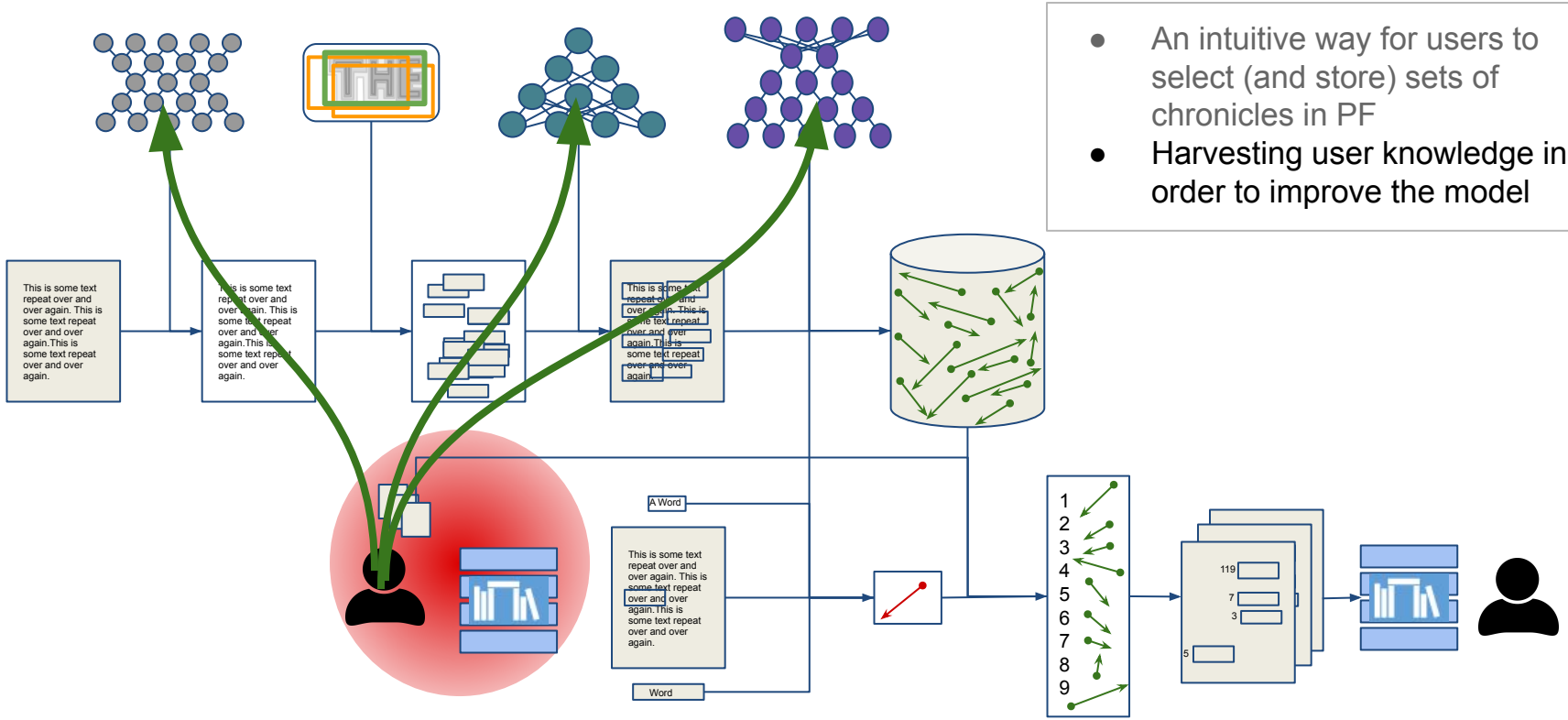
- An intuitive way for users to select (and store) sets of chronicles in PF
- Harvesting user knowledge in order to improve the model
- Repurpose the pipeline for other document types eg: Maps
- Repurpose the pipeline for other object types eg: Images/stamps/etc.

What can be next?

- An intuitive way for users to select (and store) sets of chronicles in PF

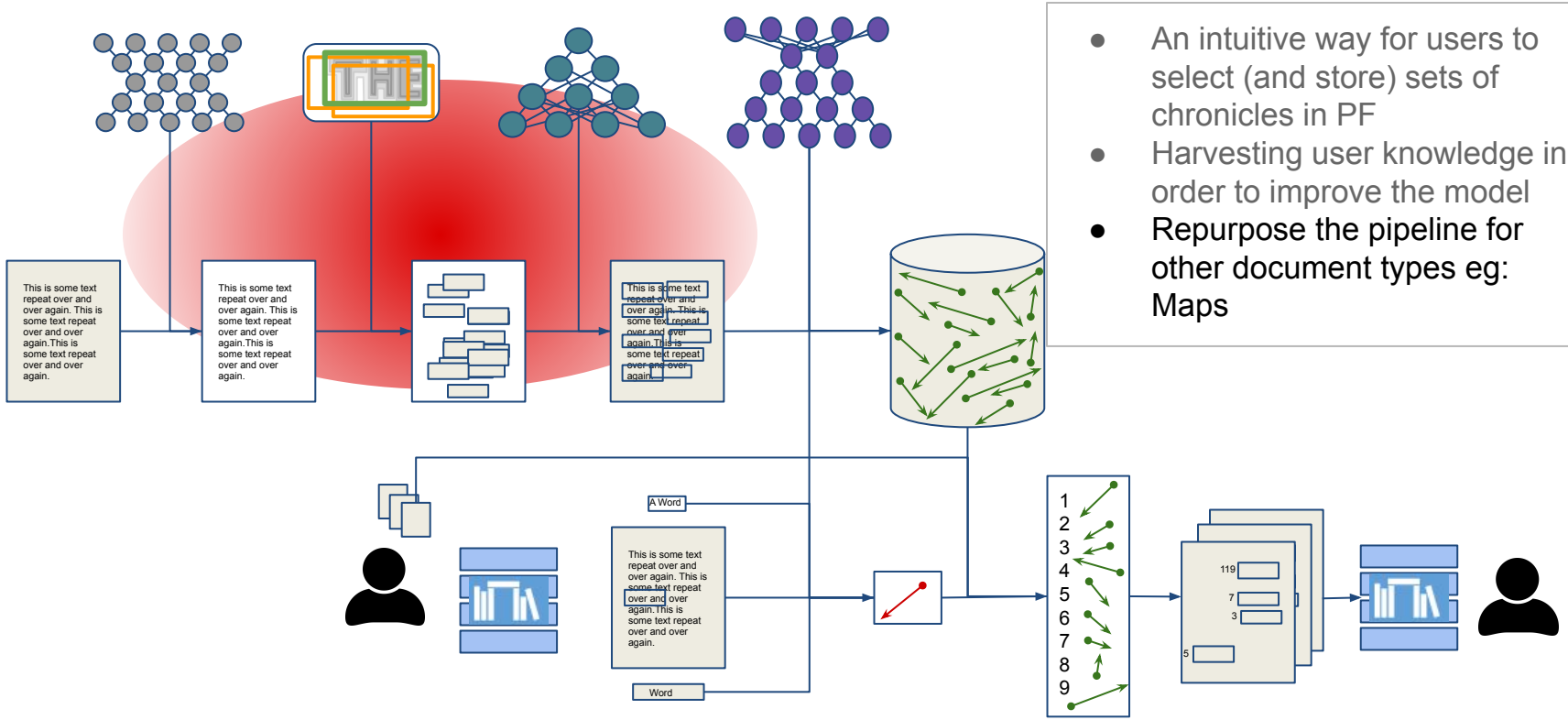


What can be next?



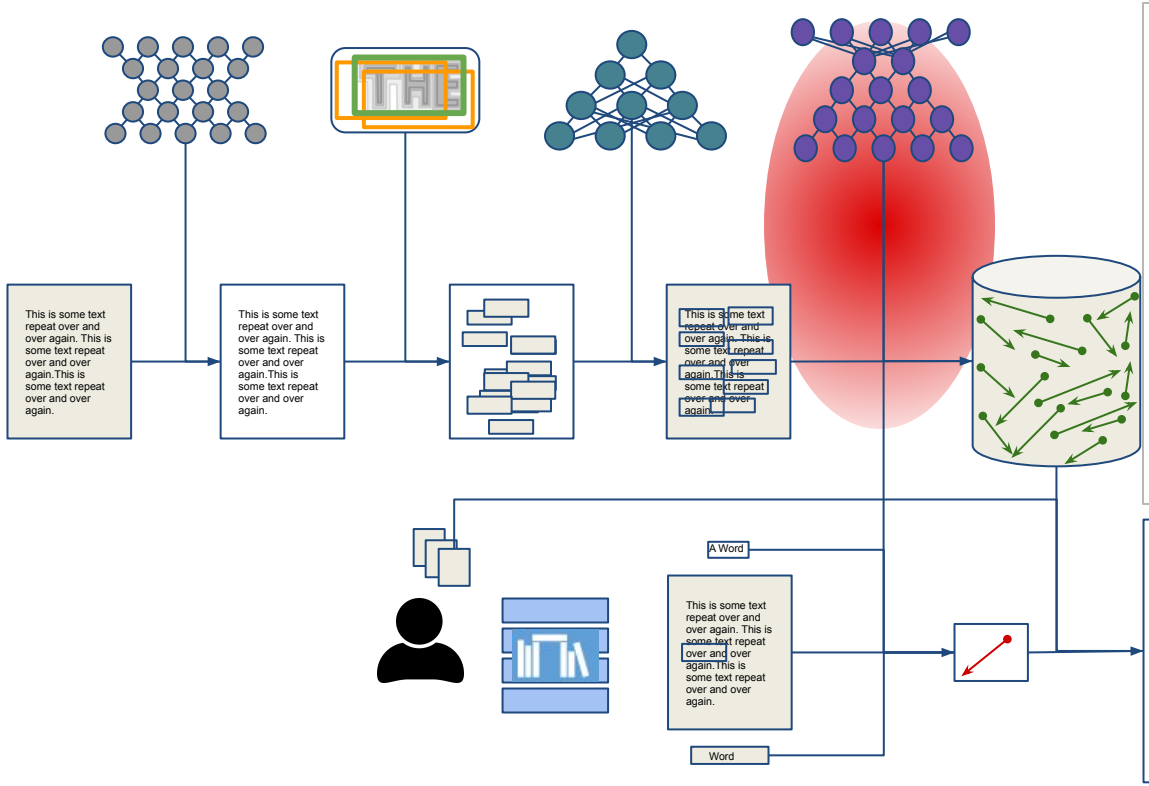
- An intuitive way for users to select (and store) sets of chronicles in PF
- Harvesting user knowledge in order to improve the model

What can be next?



- An intuitive way for users to select (and store) sets of chronicles in PF
- Harvesting user knowledge in order to improve the model
- Repurpose the pipeline for other document types eg: Maps

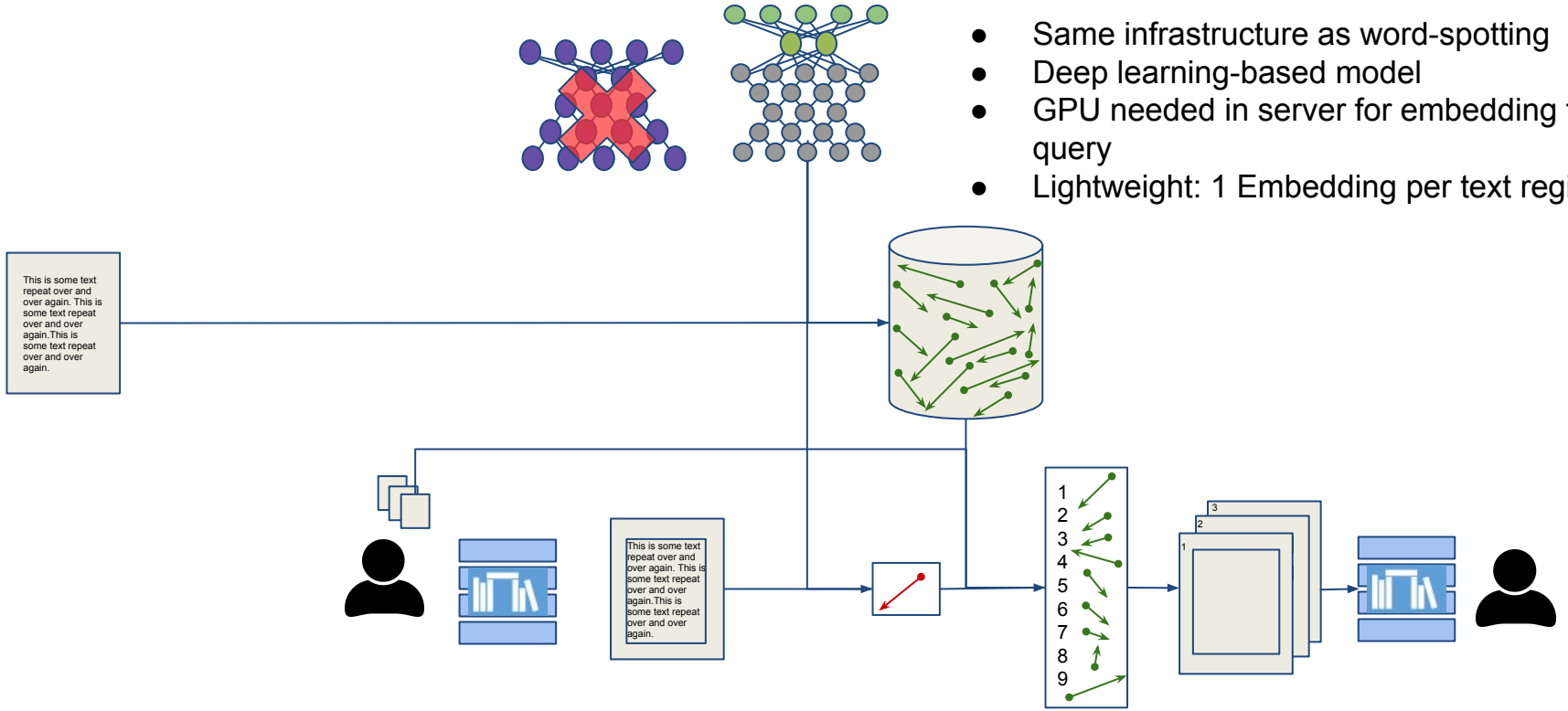
What can be next?



- An intuitive way for users to select (and store) sets of chronicles in PF
- Harvesting user knowledge in order to improve the model
- Repurpose the pipeline for other document types eg: Maps
- Repurpose the pipeline for other object types eg: Images/stamps/etc.

Questions?

Writer Identification



- Same infrastructure as word-spotting
- Deep learning-based model
- GPU needed in server for embedding the query
- Lightweight: 1 Embedding per text region