Hypernym Discovery

Course: Intro to NLP (CS7.401)

Team No.: 37
Team name: Natural Barrier

Faculty: Prof. Manish Srivastava

Mentor: Nirmal Surange, Sagar Joshi

Presenters:

- Aditya Kumar Singh
- Dhruv Srivastava
- Nayan Anand

Agenda

To share our intuitions, approaches and the results obtained for implementing the given task: Hypernym Discovery.

- Problem description
- Available Datasets
- Implementation
  - Generating embeddings from scratch and turning the task into classification! (Baseline)
  - Exploit embeddings even more
  - Exploit Hypernym Hierarchy
  - Current SoTA- CRIM
- Results (Compiled)
- Challenges faced
- Conclusion

Problem description

- Given an input term (or a hyponym **q**), our algorithm need to retrieves a ranked list (of length 15) of its suitable hypernyms.
- This is a “is-a” relation, which relates general terms to their instances or subtypes.
- This task requires the model to be successful at Natural Language Understanding (NLU).

Hyponym	Hypernyms
gasworks (concept)	plant, constructed structure, manufacturing plant, building complex, factory, manufactory
farming (concept)	economic sector, manufacture, cultivation
Emmett Tyrrell (entity)	newspaper columnist, writer, person, columnist, editorialist, media professional, journalist

Data Overview and Analysis

There are **three** primary benchmarks datasets available that can be used for hypernym discovery:

- ➔ **WordNet**: A widely used lexical knowledge base where words and phrases are connected by multiple lexical relations. It contains direct hyponym-hypernym relations available from resulting in 74,645 pairs. Available in python-nltk package.

- ➔ **BabelNet:** Derived from WikiData, it is a semantic network that connects concepts and named entities in a large network of semantic relations. BabelNet contains the missing lexical entries from resource-poor languages resulting in rich pool of hypernym relations.

- ➔ **SemEval-18 Task9:** Data is broadly classified into **General** purpose corpora and **Domain** specific corpora . Further divided into 3 sub–corporas of **English**, **Italian** and **Spanish** each for specific subtasks **1A** , **1B** and **1C** respectively as well as into **Medical** and **Music** corpus to be used for subtasks **2A** and **2B** respectively. English corpus is derived from **UMBC**, Italian from **itWac**, Spanish from (**Wikipedia**, **Europarl**, **AnCora**), medical one from the **MEDLINE**, and music one from **ELMD 2.0**.

Learning Embeddings (`word2vec`)

- After preprocessing the corpus available on SemEval 2018 Task 9 (only for English(1A), Medical(2A), and Music(2B)), we trained our **`word2vec`** model to generate embeddings of **300** dimension.

- For **Italian** and **Spanish**, we load pre-trained **`word2vec`** embeddings from open-source site.

- We exploited these trained embeddings for several upcoming approaches, where we able to make sense of each word through its vector representation.

Baseline Models

- Self-generated embeddings
- Classification using LSTM
- Classification using GRU

Training a skip-gram embedding from scratch

- Why we did this?
  Pretrained embeddings like word2vec do not reflect hyponym-hypernym relations since it groups the items that are similar to each other in some way (e.g. dogs, cats, etc will be separated from each other) whereas we wish to have an embedding space where hyponyms are closer to their hypernyms (e.g. dogs, cats, etc classes to be closer to each other being animals)
- How we did it?
  We implemented negative sampling with skip-gram to learn the embeddings in such a way that we pass a hyponym and pair-wise all it's hypernyms along with 5 negative samples each. These negative samples are selected from all the hypernyms in given-data such that there is no intersection with positive-hypernyms.

Hypernym Discovery as Classification!

- Instead the Hypernym Discovery task to Hypernym Detection
- We generate all possible [Hyponym-Hypernym] pairs from the available **SemEval-2018 Task 9** dataset.
- Models learn to classify any given Input word pairs as [Hyponym-Hypernym] pair or not a [Hyponym-Hypernym] pair.
- The testing covers if the model is able to correctly classify the relationship between a input word pair as Hyponymic /Non-Hypernymic.
- The decision of existence of Hypernymity is determined using logit score with threshold >=0.5 for Hypernymity to exist.

GRU and LSTM Architecture

GRU Architecture used

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 2, 300)            541800
_________________________________________________________________
gru_1 (GRU)                  (None, 300)               541800
_________________________________________________________________
flatten (Flatten)            (None, 300)               0
_________________________________________________________________
dense (Dense)                (None, 300)               90300
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 301
=================================================================
Total params: 1,174,201
Trainable params: 1,174,201
Non-trainable params: 0
```

LSTM Architecture Used

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 2, 300)            721200
_________________________________________________________________
lstm_1 (LSTM)                (None, 300)               721200
_________________________________________________________________
flatten_1 (Flatten)          (None, 300)               0
_________________________________________________________________
dense_2 (Dense)              (None, 300)               90300
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 301
=================================================================
Total params: 1,533,001
Trainable params: 1,533,001
Non-trainable params: 0
```

GRU and LSTM Architecture

LSTM Model Hyperparameters

- Learning Rate=0.003
- Activation = Sigmoid
- Loss = Binary Crossentropy
- Optimiser = Adam
- Epochs =10
- Batch Size =32
- Total Parameters=1,533,001
- LSTM Layers =2
- Flatten Layer=1
- Dense Layer =2

GRU Model Hyperparameters

- Learning Rate=0.003
- Activation = Sigmoid
- Loss = Binary Crossentropy
- Optimiser = Adam
- Epochs =10
- Batch Size =32
- Total Parameters=1,174,201
- GRU Layers =2
- Flatten Layer=1
- Dense Layer =2

Results - BaseLine

	1A (English) (Self-trained)	2A (Medical) (Self-trained)	2B (Music) (Self-trained)
MAP	0.411934	6.6299	12.9433
MRR	0.41193	6.6850	13.0366
P@5	0.16032	1.7621	3.9999

Exploiting the embedding space even more

- ❑ Building SenseEmbeddings using knowledge-graphs
- ❑ Define a linear transformation
- ❑ Rank candidate hypernyms

Exploiting the embedding space even more

- SenseEmbeddings?
  SenseEmbedding is a fine-grain version of word embedding by clustering different contexts for each target word. It learns a distributed vector for each sense of a word. They either define a sense as a cluster of contexts where the target word appears or define a sense based on a sense inventory. Pre-trained available from BabelNet.

- How we did it?
  Using the SenseEmbeddings, we perform clustering over all hyponyms and top-k hypernyms. Then we learn a projection transformation matrix from these clusters to hypernyms with the assumption that hyponym/hypernym pairs will occur together in same space. Using this transformation matrix, we check candidacy of every hypernym and rank them. This way we converted this problem into ranking problem and therefore filtering top-n hypernyms will be our discovered hypernyms

Exploiting the embedding space even more

- Since we decided to use SenseEmbeddings, we assume that the hypernym and hyponyms are present in same space.
- We learn a hypernym-wise linear transformation function - THIS MAKES IT RESTRICTED TO DEFINED HYPERNYMS. With these intuitions we define the transformation matrix Psi such as and learn this for each domain cluster using the third equation.

$$\vec{y}^d = \Psi \vec{x}^d,$$

$$\min_{\Psi^C} \sum_{i=1}^{|T^d|} \|\Psi^C \vec{x}_i^d - \vec{y}_i^d\|^2$$

$$\text{argmax}_{\vec{v} \in \mathcal{S}} \frac{\vec{v} \cdot \Psi^C \vec{x_j}^d}{||\vec{v}|| ||\Psi^C \vec{x_j}^d||}$$

Exploiting the embedding space even more

	1A (English) (Self-trained)	2A (Medical) (Self-trained)	2B (Music) (Self-trained)
MAP	9.7517	18.571	18.294
MRR	9.6711	19.209	18.313
P@5	2.3999	13.160	12.679

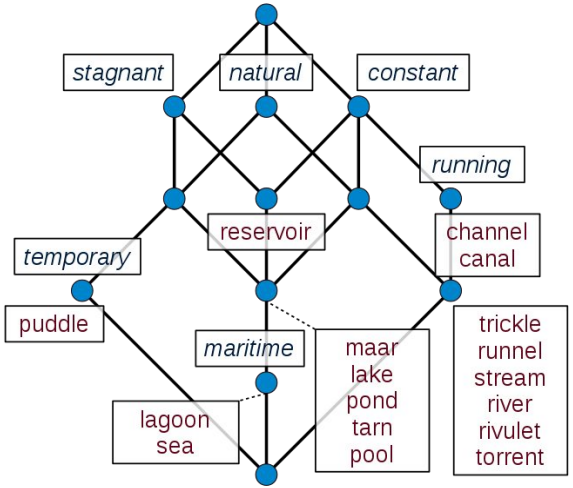
Exploiting Hierarchical properties

- Generating non-negative sparse embeddings
- Create a lattice using FCA algorithm
- Extract hierarchical features from lattice
- Train a logistic regressor and make predictions

Exploiting Hierarchical properties

- Formal Concept Analysis (FCA)
  FCA is a principled way of deriving a concept hierarchy or formal ontology from a collection of objects and their properties. For implementation purposes, the support of standard libraries was weak and therefore we used the implementation provided by Dr. Dominik Endres (2008).
- We extract some features form the generated lattice and use these features to train a logistic regression classifier.
- In the end, we take predictions from logistic regiressor and use the probabilities to filter top-n candidate hypernyms.

bodies of water	attributes					
	temporary	*running*	*natural*	*stagnant*	*constant*	*maritime*
canal		✓			✓	
channel		✓			✓	
lagoon			✓	✓	✓	✓
lake			✓	✓	✓	
maar			✓	✓	✓	
puddle	✓		✓	✓		
pond			✓	✓	✓	



Exploiting Hierarchical properties

	1A (English) (Self-trained)	2A (Medical) (Self-trained)	2B (Music) (Self-trained)
MAP	12.1396	33.7177	33.6213
MRR	26.4008	50.8266	51.1198
P@5	11.4344	32.14	32.02

CRIM

- Unsupervised Pattern-Based HD
- Supervised Projection Learning
- Hybridization

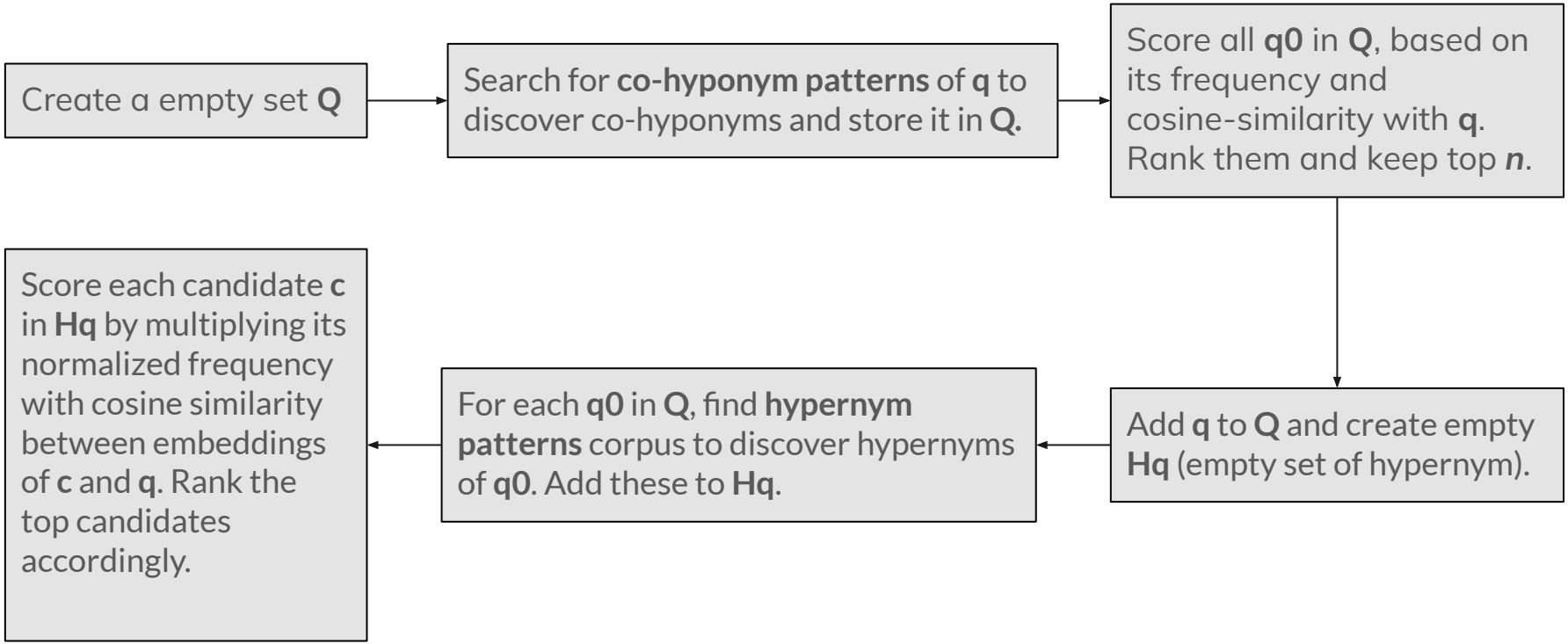
Finding Hearst-like Patterns

Dan Jurafsky

Hearst's Patterns for extracting IS-A relations

Hearst pattern	Example occurrences
X and other Y	...temples, treasuries, and other important civic buildings.
X or other Y	Bruises, wounds, broken bones or other injuries...
Y such as X	The bow lute, such as the Bambara ndang...
Such Y as X	...such authors as Herrick, Goldsmith, and Shakespeare.
Y including X	...common-law countries, including Canada and England...
Y , especially X	European countries, especially France, England, and Spain...

Finding Hearst-like Patterns - Algorithm



Note: Since embedding based, pattern-based co-hyponyms and hypernyms can find terms not included in the vocabulary. But those “query” which are OOV we simply discard them.

Learning Projection Matrix - Supervised Model

- Given a pair of hyponym and hypernym embeddings, **(eq, eh),** project **eq** with **k** different square projection matrices, **ϕi**, to obtain **k** different **d** dimensional vectors (stacked in column to give **P**).

$$P_i = (\phi_i \cdot e_q)^T \quad \phi_i \in \mathbb{R}^{d \times d} \text{ for } i \in 1, \ldots, k$$

$$P = \begin{bmatrix} \text{——} & P_1 & \text{——} \\ \text{——} & P_2 & \text{——} \\ & \vdots & \\ \text{——} & P_k & \text{——} \end{bmatrix}_{k \times d}$$

- Compute cosine similarity (dot product) of each projection with given hypernym **eh**.

$$s = P \cdot e_h \quad s \in \mathbb{R}^{k \times 1}$$

- Generate ***logits*** from linear transformed similarity scores.

$$y = \sigma(W \cdot s + b)$$

- Based on these ***logits***, select the top-ranked hypernym candidates (Our case # of hypernyms to estimate = 15).

Training SetUp - Supervised Model

- Training model using *negative sampling* with ratio of **1:10**.
- Loss function = **BCE loss**; Batch-Size = **32**; Optimizer = **Adam**(learning_rate = **1e-4**); Max-iter = **1000**; Dropout = **0.5** (between embeddings and query projections); Gradient clipping = **1e-4**; Early-Stopping Patience value = **200**.
- **24** projection matrices are used; **k = 24**.
- Samping positive examples based on their frequency — Granting higher chances to those hypernyms which occurs less often than others.

$$P(q,h) = \sqrt{\frac{\min_{h' \in D} freq(h')}{freq(h)}}$$

Hybridization (CRIM): Late Fusion of both the approaches

- Select top 100 candidates according to each hyponym, normalize their scores and sum them.
- Rerank the candidates according to this new score.
- Top 15 hypernyms against given hyponym is what expected.

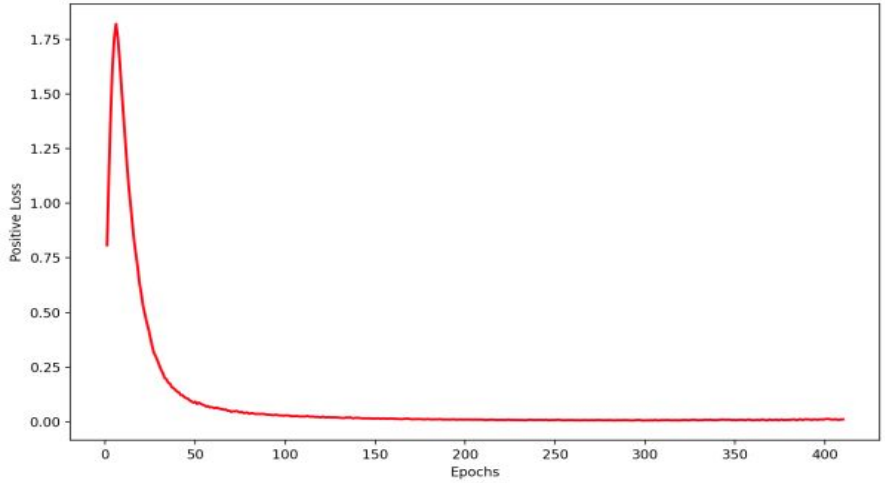
Experiments and Results - CRIM

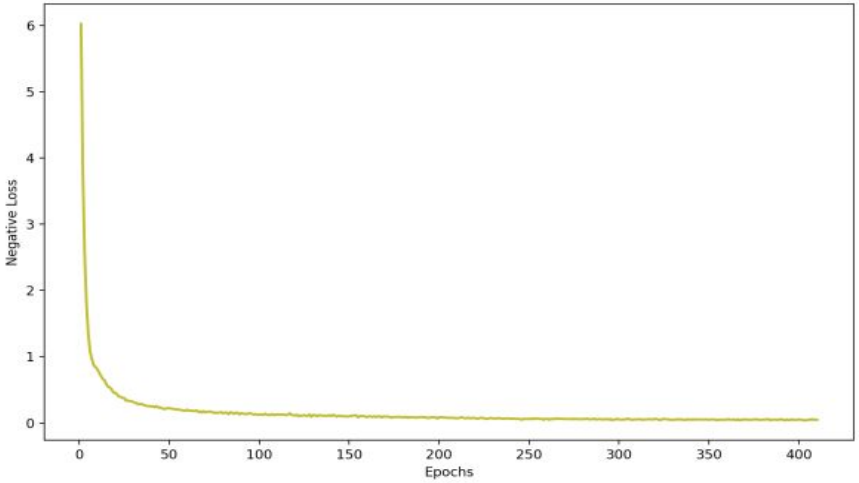
	1A (English) (Self-trained)	1B (ItWiki) (Pre-Trained)	1C (EsWiki) (Pre-Trained)	2A (Medical) (Self-trained)	2B (Music) (Self-trained)
MAP	23.53	21.37	33.41	33.80	46.20
MRR	34.44	34.29	52.23	42.61	62.39
P@5	20.43	17.44	29.23	35.16	47.85

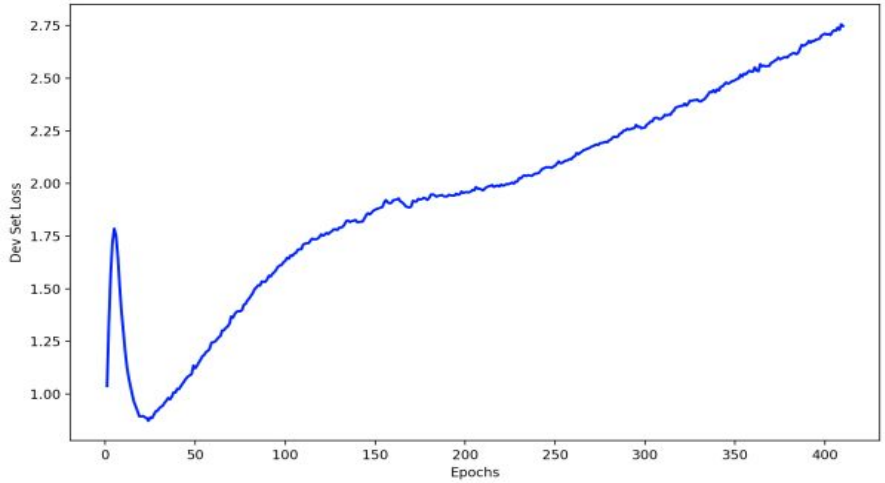
Results with different settings and embeddings.

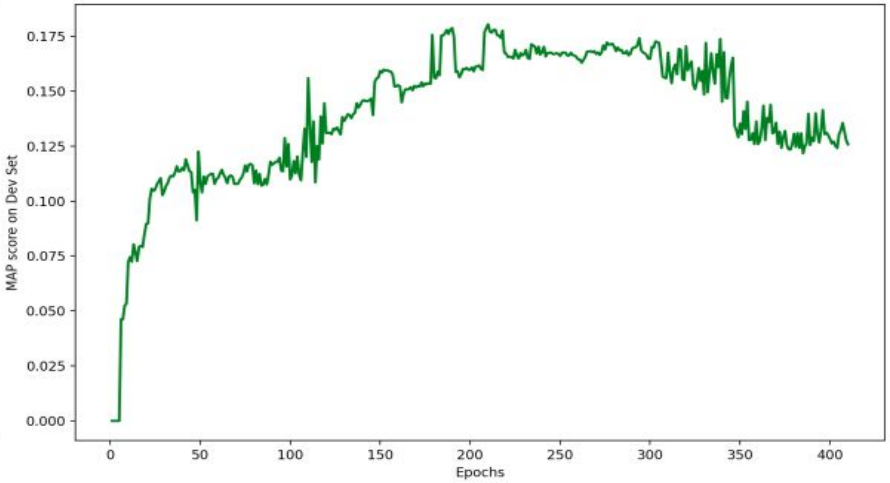
	1A (without embedding normalization)	1A (with BS=64)	1B (itWac with 128 emb size)	1C (Spanish Emb from SBWC Corpus)	2B (without emb normalization)
MAP	17.55	23.04	18.06	38.75	45.98
MRR	32.40	34.04	27.33	21.92	60.56
P@5	16.99	19.67	15.0	19.10	46.94

Plots - CRIM (1A)

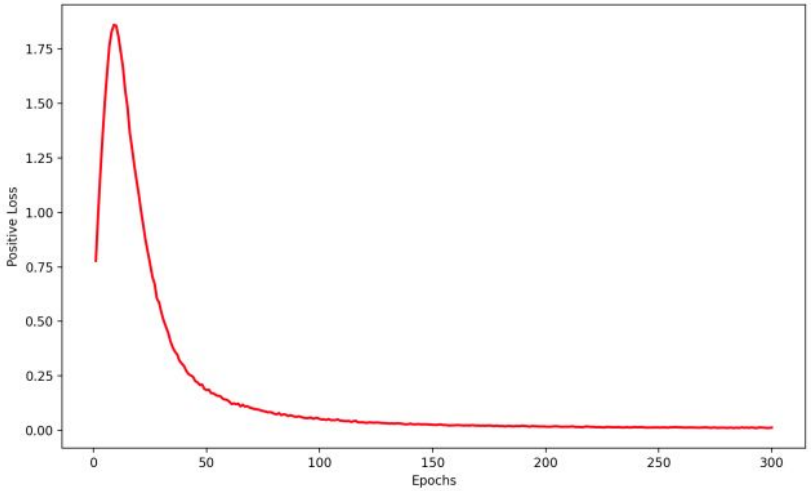


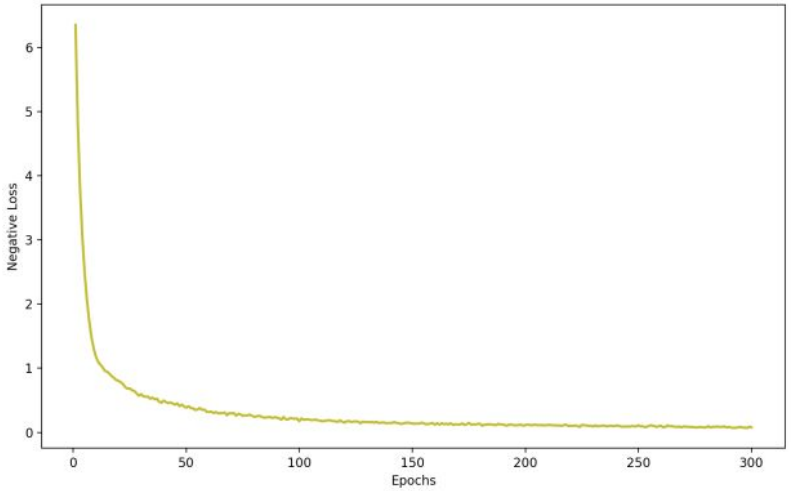


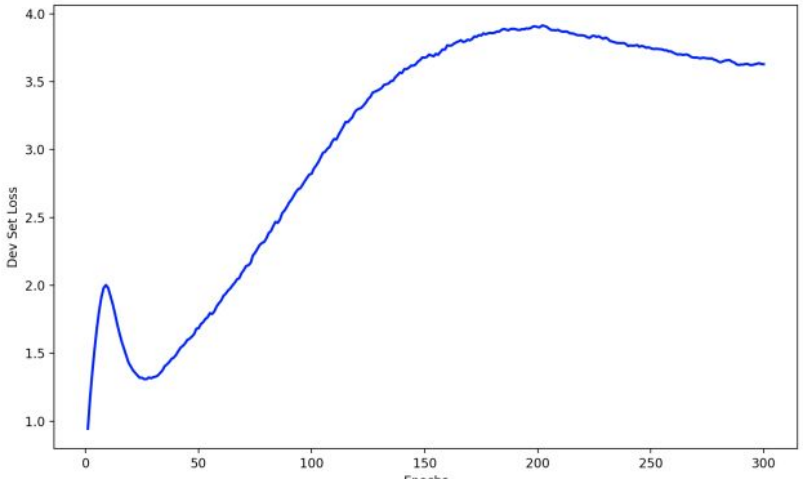


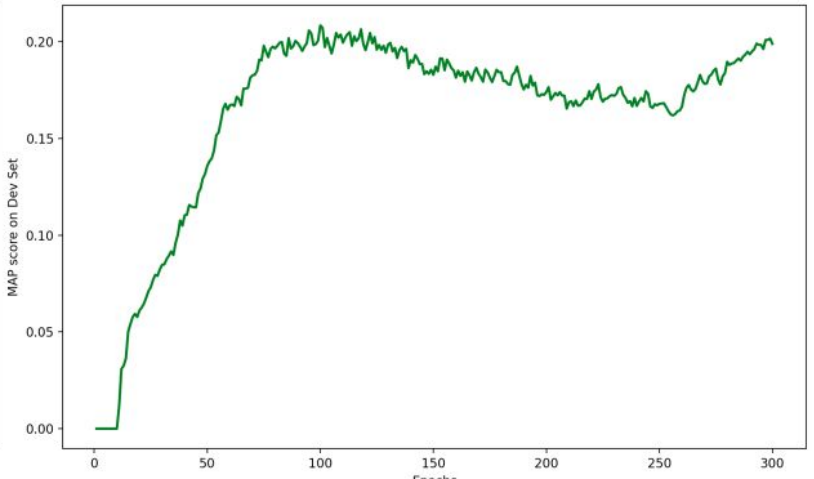


Plots - CRIM (1B)

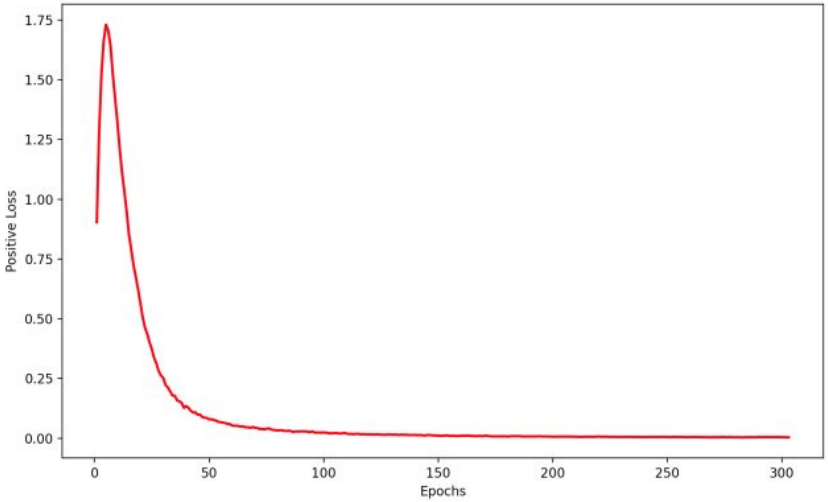


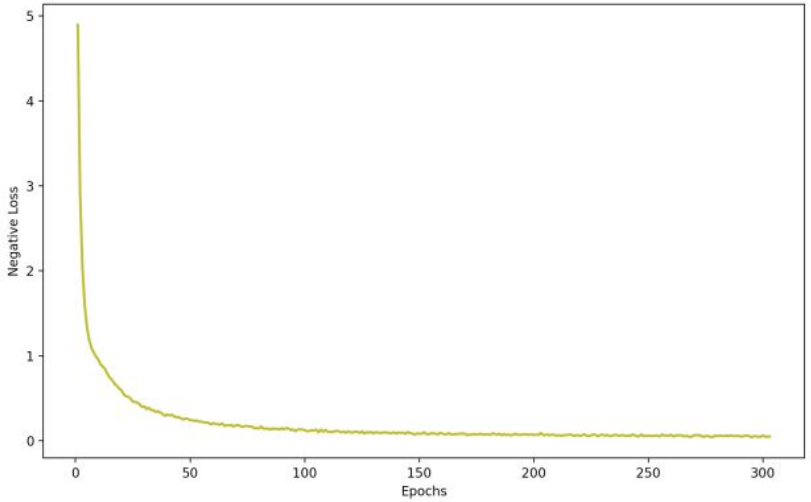


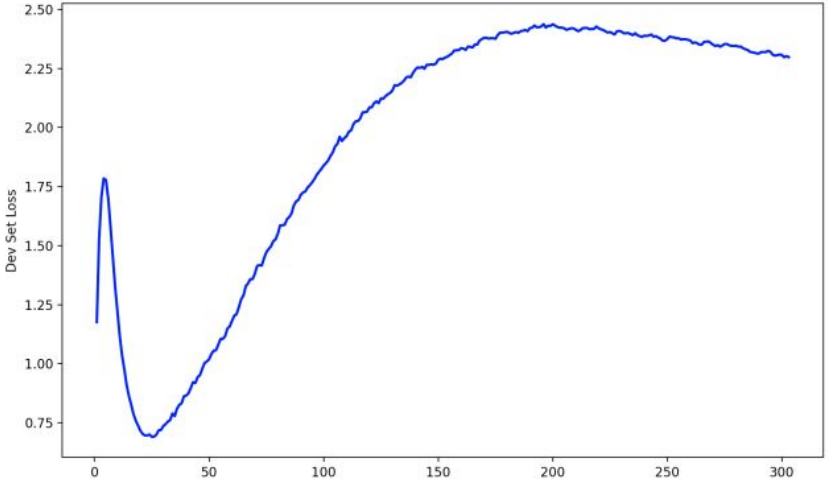


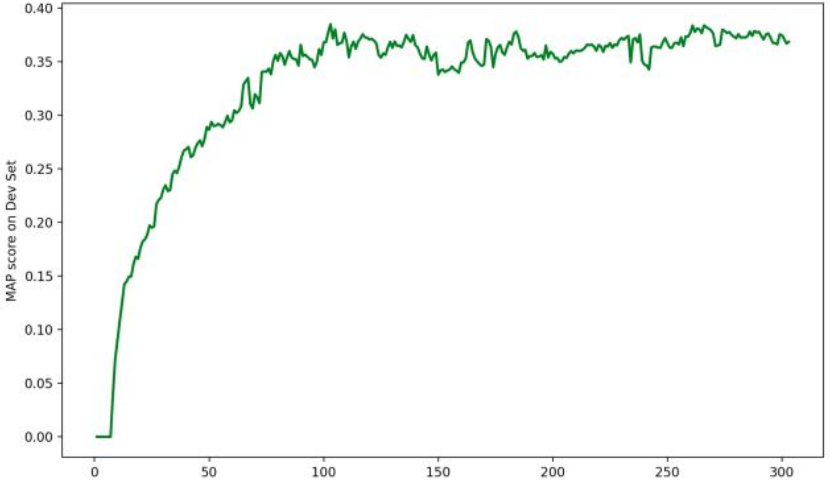


Plots - CRIM (1C)

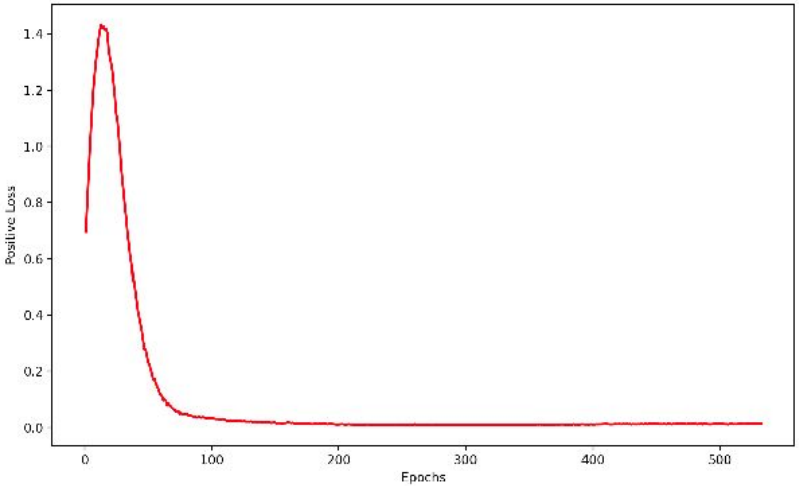


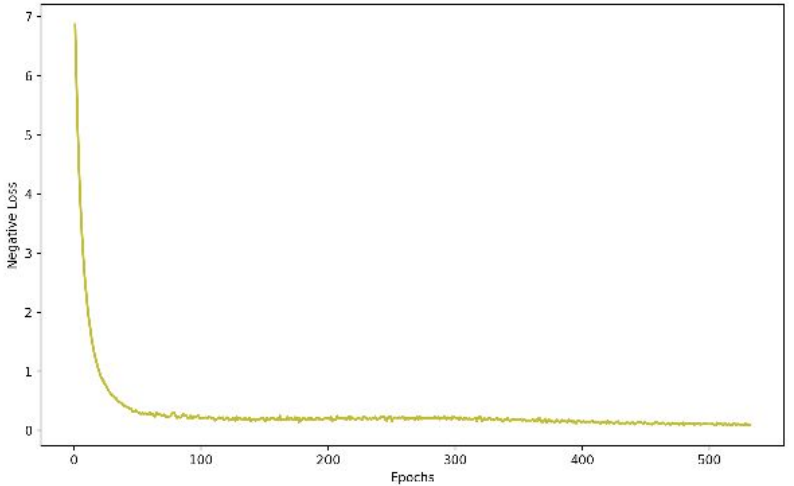


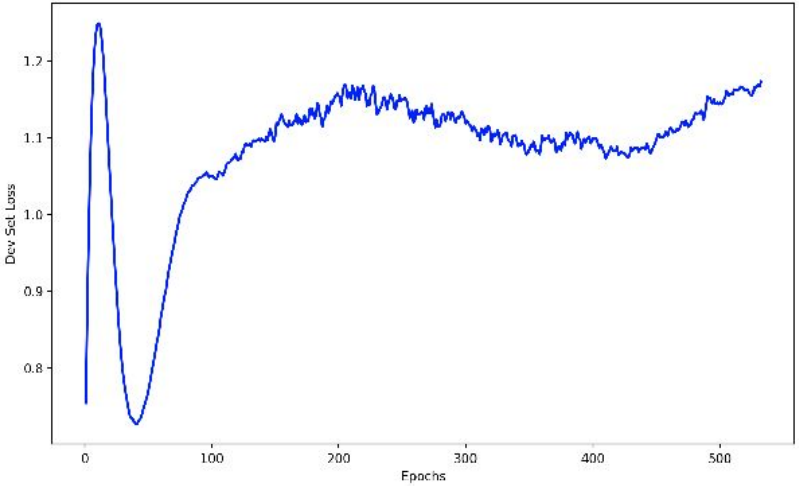


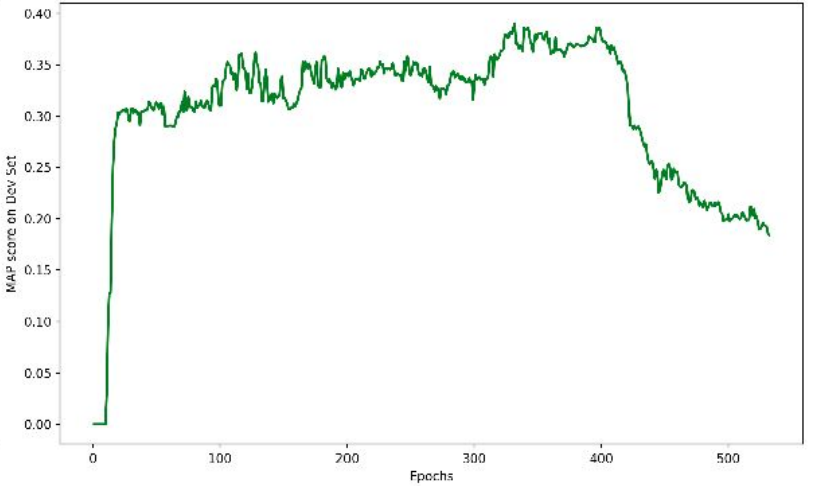


Plots - CRIM (2A)

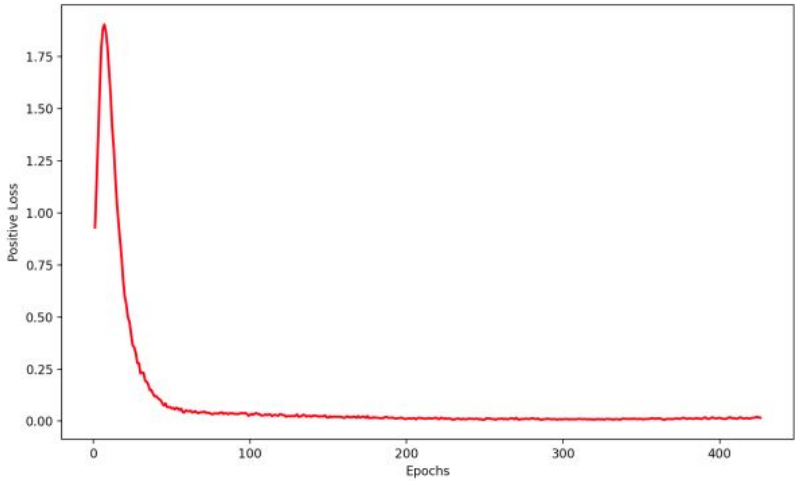


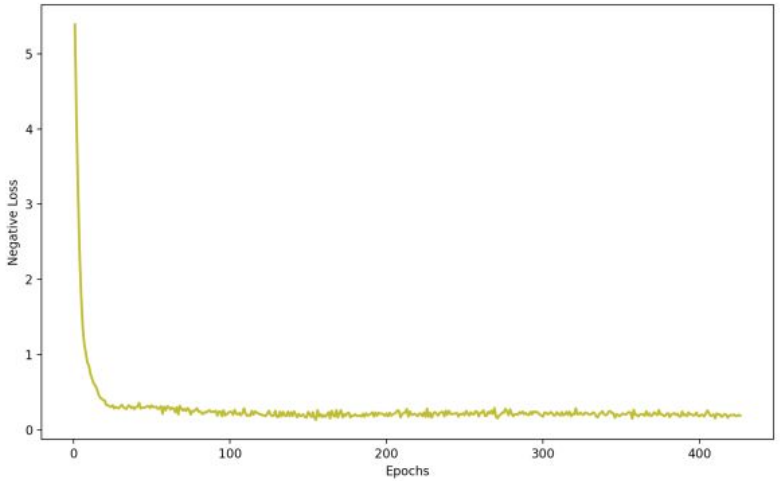


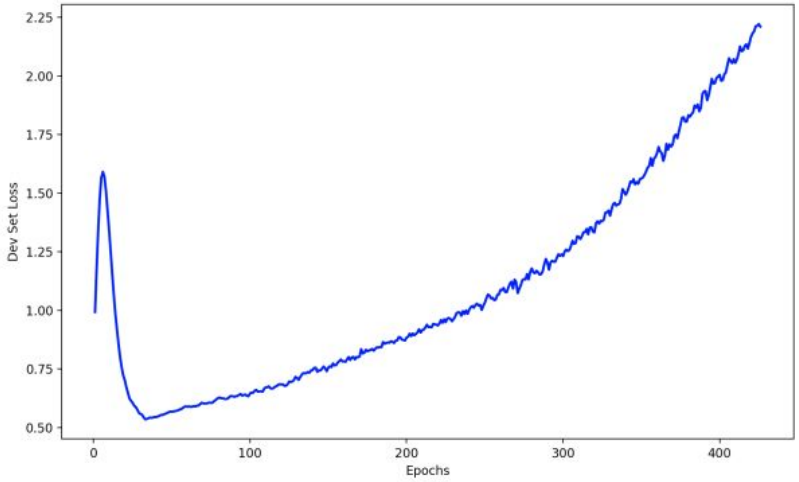


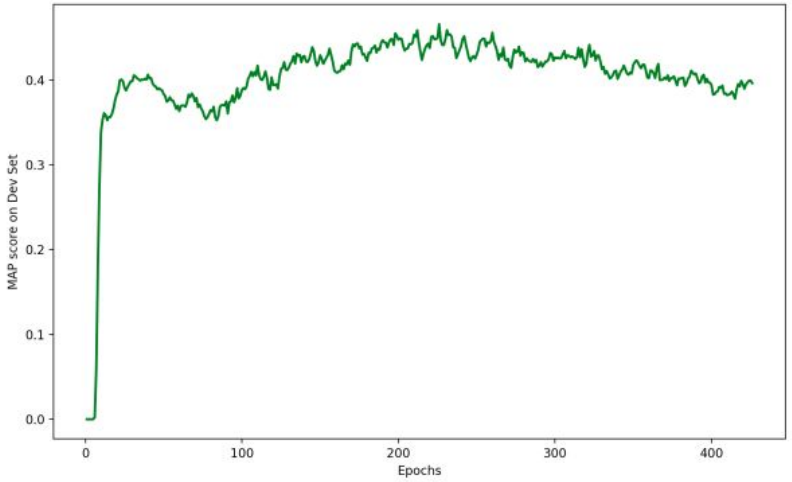


Plots - CRIM (2B)



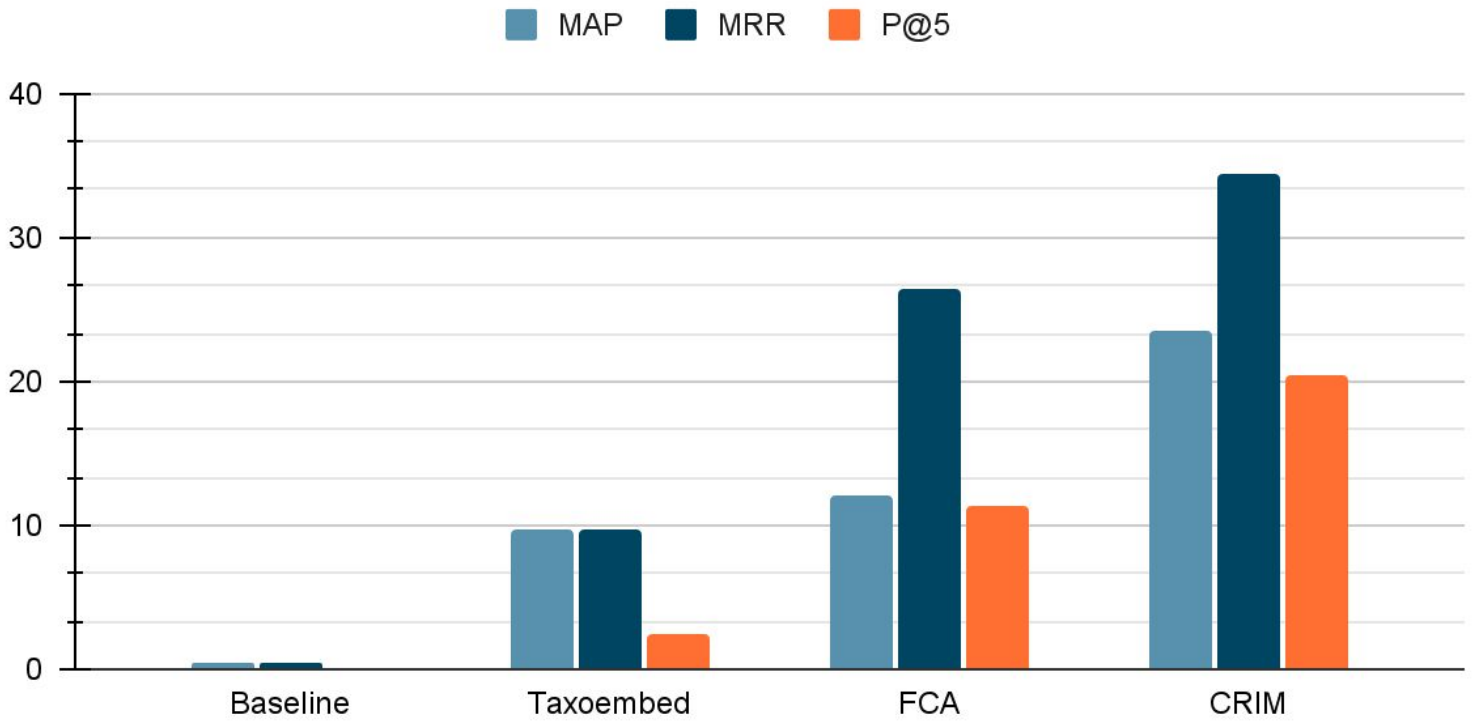




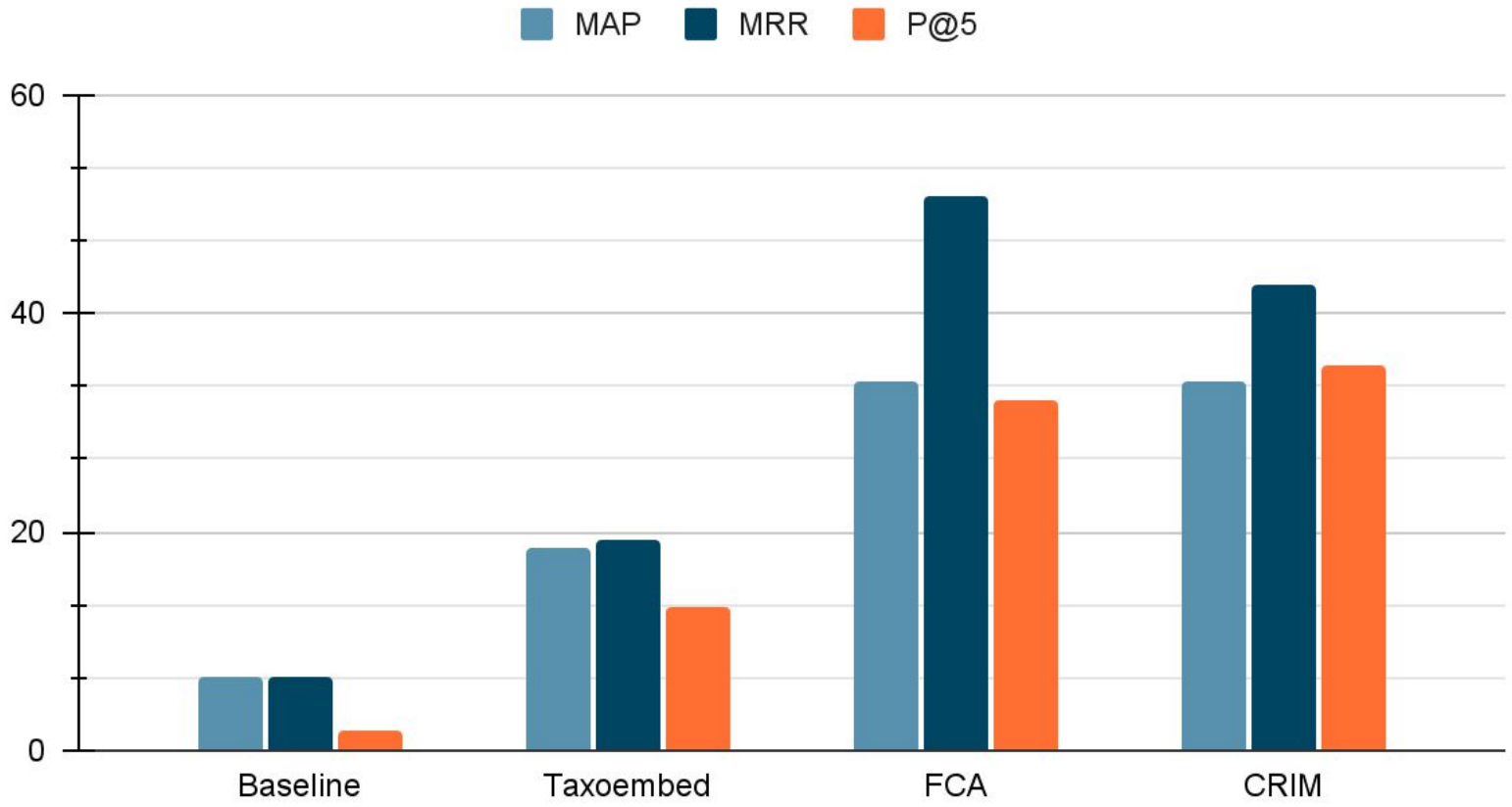


Results Summary

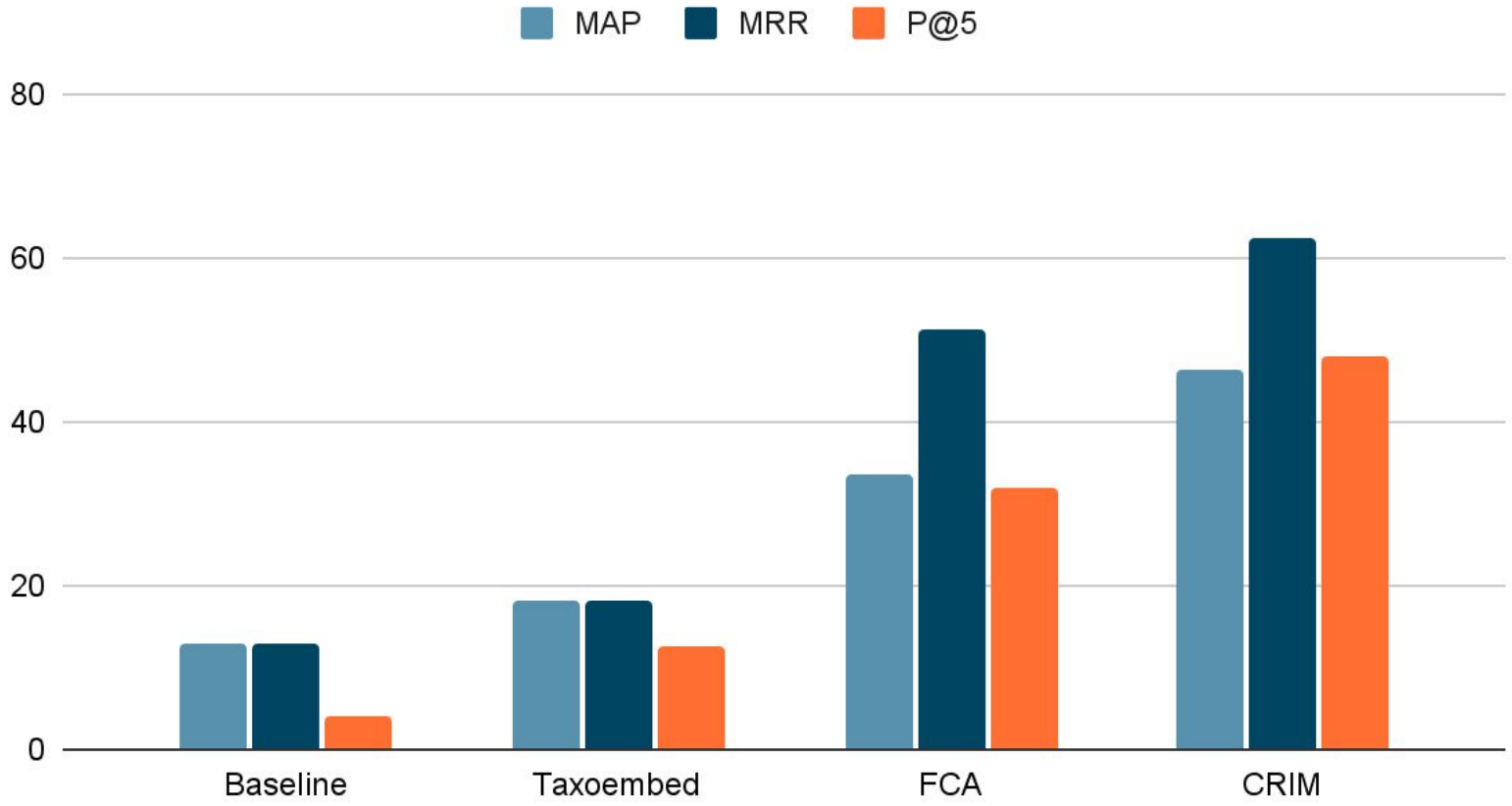
1A (English)



2A (Medical)



2B (Music)



Qualitative Analysis

- **TaxoEmbed**: Novelty lies in the fact that it exploits distributional information and does not make use of predefined syntactic heuristics, suggesting that the information it provides and the rule-based comparison systems may be complementary.

- **FCA**
  - Applied “candidate filtering” to speed up FCA. Degrades score if turned off.
  - Obtained best results on “Spanish” and “Italian”.

- **CRIM**
  - Able to handle low-frequency queries, but couldn’t investigate how sensitive it is to term frequency.
  - Lexical memorization can sometimes be observed (e.g., “person”is the most frequent hypernym in the **1A** training data, and the model often predicts this candidate).
  - It can discover hypernyms of different senses of the same query (e.g.”aquamarine”, for which the top 15 predictions contain the valid hypernyms “spectral color” and “primary color”), and it sometimes discovers hypernyms for senses that are not represented in the gold standard (e.g. there is a college named “Swarthmore”, and “hypostasis” has senses related to linguistics and philosophy).

Conclusion...

- **FCA** experimented with the integration of sparse word representations into the task of hypernymy discovery. Utilized it in two ways, i.e. via **building concept lattices using formal concept analysis** and **modeling the hypernymy relation with the help of interaction terms**.
- **CRIM** Combines a novel supervised projection learning algorithm and an unsupervised pattern-based algorithm which exploits co-hyponyms in its search for hypernyms.
- We provided a comprehensive overview of Hypernym Discovery through several experimentation ranging from traditional to deep-learning based SoTA methods.
- We also tried to invoke **Skip-Gram** embeddings into the **CRIM** model to check for any additional increment. And to our surprise we got an increment of **12%**.

References

- Hypernym Discovery via a Recurrent Mapping Model
- CRIM at SemEval-2018 Task 9 :A Hybrid Approach to Hypernym Discovery
- Automatic Acquisition of Hyponyms from Large Text Corpora
- Efficient Estimation of Word Representations in Vector Space
- Distributional Hypernym Generation by Jointly Learning Clusters and Projections

Thank You