



HYPERNYM DISCOVERY

FINAL REPORT

COURSE CODE: INTRO TO NLP - CS7.401.S22

Advisor:

Prof. Manish Shrivastava

Team Name:

Natural Barrier

Project Mentor:

Nirmal Surange

Project Representatives:

Aditya Kumar Singh - 2021701010

Dhruv Srivastava - 2021701021

Nayan Anand - 2021701014

Academic year:

2021-2022

Contents

1	Project Description	3
1.1	Problem statement	3
1.2	Overview of upcoming sections	3
2	Datasets	3
2.1	WordNet dataset	3
2.2	BabelNet	3
2.3	SemEval2016-Task13	4
2.4	SemEval2018-Task9	4
2.5	Taxonomy Extraction Evaluation (TExEval-2)	5
3	Project Implementation	5
3.1	Baseline Idea	5
3.1.1	Dataset Preparation	5
3.1.2	Baseline Training and Evaluation	5
3.2	Exploiting the embedding space	5
3.2.1	Defining word embedding	6
3.2.2	Learning the hypernym transformation matrix	6
3.3	Exploiting the hierarchical properties of hypernyms	7
3.3.1	Generating the sparse word vectors	7
3.3.2	Using Formal Concept Analysis to exploit the hierarchical nature of hypernyms	7
3.3.3	Feature generation	8
3.3.4	Logistic Regression for final classification	8
3.4	CRIM	8
3.4.1	Unsupervised Pattern-Based Hypernym Discovery	8
3.4.2	Supervised Projection Learning	9
3.4.3	Hybridization: How Supervised and Unsupervised are fused?	11
4	Experiments and Results	11
4.1	Learning Word Embeddings	12
4.2	Results & Plots	12
4.2.1	Plots	13
5	Conclusion	16

1. Project Description

1.1. Problem statement

Hypernymity shows the relationship between a generic term (hypernym) and a specific instance of it (hyponym) [7]. A hypernym, also known as a superordinate, has a larger semantic range than a hyponym. To understand this better let's look at an example. "From the toolbox, can you please pass a wrench, said the plumber to his assistant". In the above sentence, "tool" is the hypernym for "wrench". Here it is worth noting that the relationship between a *Hypernym-Hyponym* pair is asymmetric i.e., hyponyms can always be replaced with a Hypernym in a sentence whereas the vice versa might not always be possible.

Multiple works in History showcase the task of identifying hypernymic relation for a given pair of candidate terms [4]. Out of the varied approaches applied, the binary classification approach has been one of the most popular approaches.

In this project we look forward to explore, multiple strategies for identifying the existence of Hypernymity, that includes replicating the experiments present in literature for this task and adding a few experiments of our own as an attempt to further improve the results.

1.2. Overview of upcoming sections

The following sections in the report outlines the progress made thus far. We have made attempts in multiple directions, not limited to unsupervised pattern-based approach, supervised projection learning approach, as well as the Hybrid approach (combining both supervised and unsupervised) for hypernym discovery which in turn has built up on top of discovering the Hypernyms using Hearst pattern[13]. Section 2 explains the dataset explored and

2. Datasets

2.1. WordNet dataset

WordNet is a vast english lexical database which consists of cognitive synonyms (synsets), each expressing a distinct concept which are interlinked by conceptual-semantic and lexical relations. Synsets are nothing but synonymy in which synonyms' meanings are so similar that they can't be distinguished denotatively or connotatively.

A modest number of "conceptual relations" connect each of WordNet's 117,000 synsets to other synsets. There are as many synsets as there are word forms with different meanings in WordNet. Hence, every form-meaning pair in WordNet is distinct. This structure makes WordNet a useful tool for computational linguistics and natural language processing due.

The primary version of WordNet is publicly available under a BSD style license and it has multiple derivatives for other languages. The last stable version 3.1 (June 2011) is incorporated in the famous open-source Python library- Natural Language Toolkit (NLTK). Fig. 1 represents the hierarchy of words in WordNet data-set.

2.2. BabelNet

BabelNet is a multilingual encyclopedic dictionary with lexicographic and encyclopedic coverage of terms, as well as an ontology that connects concepts and named entities in a massive network of semantic relations termed Babel synsets, which is made up of 13,801,844 million nodes. It covers 500 languages and contains almost 20 million synets with over 1.4 billion word sense. Each Babel synset represents a certain meaning and comprises all of the synonyms that communicate that meaning in multiple languages. Fig. 2 represents the sources used to build babel data set.

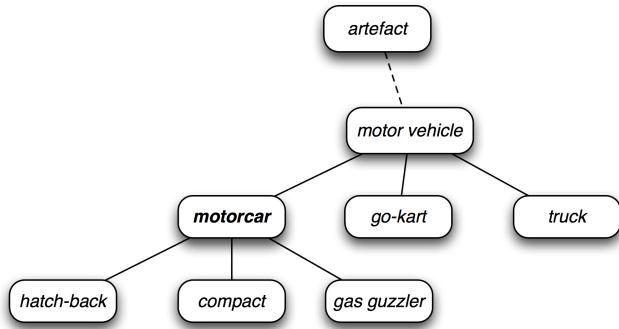


Figure 1: Hierarchical structure of WordNet dataset.

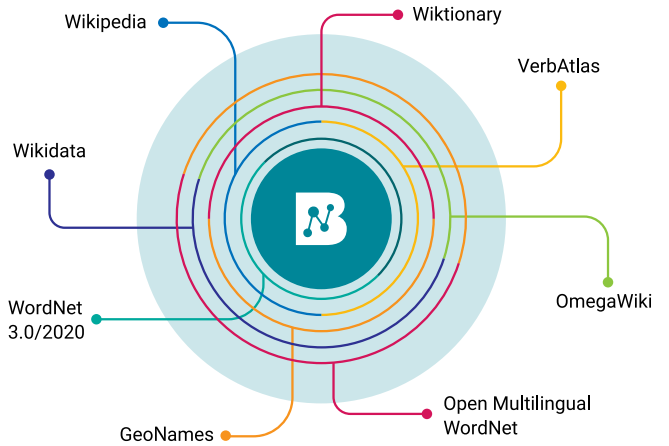


Figure 2: Different sources used to build BabelNet data-set

2.3. SemEval2016-Task13

The dataset is sourced from three target domains; Environment, Food and Science. English and Multilingual taxonomies were sourced from Eurovoc1, a large lexical database of English, WordNet, and a general purpose resource, the Wikipedia Bitaxonomy along with several other domain specific resources. While the English Taxonomies were sourced from the above mentioned resources, the Multilingual Taxonomies were manually translated from English to Dutch , French and Italian by linguists.

2.4. SemEval2018-Task9

The dataset provided for the Hypernym discovery task of SemEval 2018 is broadly divided into two categories : (A) General Purpose Corpora and (B) Domain Specific Corpora.

(A) General Purpose Corpora: The General purpose corpora is again further subdivided into 3 sub –corpo-ras of English,Italian and Spanish each for Specific subtasks 1A , 1B and 1C respectively. The English corpus is a subset of UMBC corpus (a resource composed of paragraphs extracted from the web as part of the Stanford WebBase Project) whereas the Italian and Spanish corpus were constituted from itWac corpus which in itself has been extracted from different sources of the web within the .it domain and 1.8-billion-word Spanish corpus, which contains heterogeneous documents from different sources respectively.

(B) Domain Specific Corpora: Domain Specific Corpora is further subdivided as well into Medical and Music Corpus to be used for Subtasks 2A and 2B respectively. The medical corpus is derived from the MED-LINE9 (Medical Literature Analysis and Retrieval System) repository whereas the music corpus is sourced from a concatenation of several music-specific corpora, articles from the music branch of Wikipedia, and a corpus of album customer reviews from Amazon.

Some of the samples from this data-set are-

Hyponym	Hypernym	Corpus
sodium ni-trite	chemical series bond, inorganic compound, chemical bond, chemical group	English
hymn	religious song, vocal, song, track, human voice church music, religious music musical composition, piece of music, opus, composition, christian music, musical style, music genre musical work, work of art	Music
childhood obesity	malady, illness, sickness, disorder, disease	Medical
cantinero	camarero, barman, trabajador, empleado, persona	Spanish
gettone	dischetto, moneta, compenso, disco, cerchio	Italian

Table 1: Samples of Hyponyms and corresponding Hypernyms from SemEval 2018 Task 9 data-set.

All the corporas are further subdivided into train , test and validation splits, which in turn is sub-divided

into data and gold splits. The data split contains the Hyponyms which have 2 subcategories namely - Entity (names, location, etc) and Concepts (phenomenon, activity, etc). The Gold split comprises of the Hyperyms corresponding to each Hyponym present in the data split.

2.5. Taxonomy Extraction Evaluation (TExEval-2)

Taxonomies were evaluated through comparison with gold standard relations collected from WordNet and other well known, publicly available taxonomies. Relationships with gold standards were derived from manually constructed taxonomies, classification schemes, and/or ontologies, depending on the domain. In addition, the gold standard was then supplemented by manual evaluations of relations that are not covered by it, as well as quantitative and qualitative structural analyses of the resulting graph. Among the structural criteria, the presence of cycles, the number of intermediate nodes in comparison with the number of leaf nodes, and the number of overgeneric relationships with the root node was taken into consideration. A standard precision, recall, and F1 measure was used to evaluate submitted relations between terms.

3. Project Implementation

3.1. Baseline Idea

We choose two baseline Models, LSTM and GRU. We redefine the problem from discovering Hypernyms given a Hyponym to selecting a word pair and identifying Hypernymic relations between them. This way we not only save a lot of compute time but also reduce our output vector to be of 1×1 from $1 \times \text{vocabulary_size}$ for each Hypernym.

3.1.1 Dataset Preparation

The original dataset consists of “data” folder that contains Hyponyms and a “gold” folder that contains the Hypernyms for the corresponding Hyponyms. We select all these Hyponym–Hypernym pairs and associate a corresponding training label 1 for all such pairs. These pairs are to be used as training data for our baseline models.

To overcome this problem we introduce negative samples in our training data for baseline models. For each correct Hyponym–Hypernym pair the model sees we introduce 5 negative word pairs which do not have any hypernymic relationship between them. This way we ensure that the model actually learns to identify hypernymity and doesn’t randomly make predictions.

3.1.2 Baseline Training and Evaluation

For both our Baseline models **(A)** GRU based model and **(B)** LSTM based model the embedding was learnt using **Word2Vec** on specific parent corpuses from which the **Semeval 2018 Task 9: Hypernym Discovery** dataset was sourced. We then provide the embedding in form of a vector of size (2×300) that consists of embeddings of [Hyponym–Hypernym pair] each of size (1×300) . This is done for all such Hypernym–Hyponym pairs designed to train the baseline models. We also have a validation set designed, which shares similar structure as that of the training set.

For Evaluation of these models we come up with a test set design that consists of each candidate Hyponym with all words in the vocabulary as the input pair combinations. The embeddings of these input pairs are extracted from the pretrained embeddings. These embedding pairs are then given to the model as input and model assigns a Binary Cross entropy score for each such input pair. We discard any score below 0.5 as an indicative on non-existence of Hypernymity for the input pair. Amongst the remaining pair we pick top 15 pairs for candidate Hyponym in terms of Binary cross entropy score or top X pairs in case we get X qualified word pairs per candidate Hyponym (where $X < 15$). The results are described in detail in the section 4.2.

3.2. Exploiting the embedding space

For hypernym discovery task, most of the methods rely on is-a hypernymic relations as their backbone for analysis. This leaves an opportunity to exploit the embedding space where we can use knowledge-base embeddings,

specifically SenseEmbeddings for hypernym analysis. Here, we attempt to discover the hypernymic relations by exploiting linear transformations in embedding space.

3.2.1 Defining word embedding

We define sense vectors which constitutes the sense embedding space that we use for training our hypernym discovery algorithm. Vectors in sense-space are latent continuous representations of word sense derived from Word2Vec architecture. Following steps are applied to construct the sense embedding-

1. **Clustering individual word embedding into domain clusters \mathcal{C} .** Using k-means clustering algorithm, we cluster individual word's embedding into different domains. Then learn the cluster-wise linear projection over all pairs of the expanded training set. The aim is to learn a function sensitive to a pre-defined knowledge domain, under the assumption that vectors clustered with this criterion are likely to exhibit similar semantic properties.
2. **Generating a lexical domain vector** For every domain (hyponym here), we build a lexical-vector which is concatenation of all the hypernyms belonging to the cluster of that domain. Now using the pre-trained domain vectors (originally from BabelNet but here we used Word2Vec) as targets, we calculate the similarity scores and assign every cluster a domain based on highest similarity score.

$$\hat{d} = \max_{d \in \mathcal{D}} WO(\vec{d}, \vec{b})$$

Here, \vec{d} is our lexical-vector, \vec{b} is the target domain vector and WO refers to the *Weighted Overlap* comparison measure which is defined as-

$$WO(\vec{v}_1, \vec{v}_2) = \sqrt{\frac{\sum_{w \in \mathcal{O}} (rank_{w, \vec{v}_1} + rank_{w, \vec{v}_2})^{-1}}{\sum_{i=1}^{|\mathcal{O}|} (2_i)^{-1}}}$$

Where, $rank_{w, \vec{v}_2}$ is the rank of word w in the vector \vec{v}_i according to it's weight, and \mathcal{O} is the set of overlapping words between the two vectors.

To maintain the reliability of the domain labels, those synsets whose maximum similarity score is below a certain threshold are not annotated with any domain.

3. **Preparing the data** After assigning domains to each candidate, we create our data by taking individual hypernyms and domains as pairs (domain, hypernym) from the domain-all_hyponym mappings.

3.2.2 Learning the hypernym transformation matrix

Instead of learning a global linear transformation function over broad relations, we learned a function sensitive to a given domain. This has an important implication- the data gets restricted to defined domains we used for clustering. So, for each domain-wise expanded training set T^d , we construct a hyponym matrix $X^d = [\vec{x}_1 \dots \vec{x}_d]$ and a hypernym matrix $Y^d = [\vec{y}_1 \dots \vec{y}_d]$ which are composed by corresponding SenseEmbed vectors of training pairs $(x_i^d, y_i^d) \in C_d \times C_d, 0 \leq i \leq n$.

Under the intuition that there exists a matrix Ψ so that $\vec{y}^d = \Psi \vec{x}^d$, we learn a transformation matrix for each domain cluster C_d by minimizing

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

Then for any unseen term x^d , we obtain a ranked list of the most likely hypernyms of its lexicalization vectors \vec{x}_j^d , using as measure cosine similarity.

$$\operatorname{argmax}_{\vec{v} \in S} \frac{\vec{v} \cdot \Psi^C \vec{x}_j^d}{\|\vec{v}\| \cdot \|\Psi^C \vec{x}_j^d\|}$$

This associates with each sense vector a rank which can be used to extract the top-n suitable hypernyms for our task. The ranking function is defined as-

$$\lambda(\vec{v}) = \frac{\cos(\vec{v}, \Psi^C \vec{x}^d)}{\operatorname{rank}(\vec{v})}$$

where $\operatorname{rank}(\vec{v})$ is the rank of \vec{v} according to its cosine similarity with $\Psi^C \vec{x}^d$.

This poses the hypernym discover task as a ranking problem and using the top-k filters, we can extract the most likely candidates to be the hypernyms.

3.3. Exploiting the hierarchical properties of hypernyms

Sparse word vectors can help in focusing on most salient parts of the word representations. Use of sparse, over complete representations have been motivated in various domains as a way to increase separability [5, 17].

Non-negativity as also been argued to be advantageous for interpretability [1, 10, 11] with the sense that it is advantageous to point out the relevant positive features instead of providing redundant negative ones.

Formally, we considered an attribute pair $\langle i, j \rangle \in \phi(q) \times \phi(h)$ where q is the query word and h is the hypernym candidate and $\phi(w)$ is the index of a non-zero component in the sparse representations of word w .

To better exploit the hierarchical nature of hypernyms using the non-negative nature sparse word vectors, we chose Formal Concept Analysis (FCA) algorithm which is a principled way of deriving a concept hierarchy from a collection of objects and their properties.

3.3.1 Generating the sparse word vectors

We extracted the word embedding from word2vec by training it over English (UMBC corpus [12]), Italian (Itwac corpus [3]), Spanish (SpanishWiki corpus [8]), Medical terms corpus [15] and Music corpus [18] separately. This generated the dense embedding and therefore we converted them to sparse using equation 1

$$\min_{D \in \mathcal{C}, \alpha \in \mathbb{R}_{\geq 0}} \|D\alpha - W\|_F + \lambda \|\alpha\|_1 \quad (1)$$

Here, $W \in \mathbb{R}^{d \times |V_x|}$, V_x is the size of the vocabulary and d is the size of individual embedding (set to 100). \mathcal{C} refers to the convex set of $\mathbb{R}^{d \times k}$ matrices consisting of d -dimensional columns vectors with norm at most 1, and α contains the sparse coefficients for the elements for the elements of the vocabulary. α ensures the non-negativity constraint.

3.3.2 Using Formal Concept Analysis to exploit the hierarchical nature of hypernyms

For FCA algorithm, each concept in the hierarchy represents the objects sharing some set of properties and each sub-concept in the hierarchy represents a subset of objects above it in the concept-hierarchy. For implementation purposes, the support of standard libraries was weak and therefore we used the implementation provided by Dr. Dominik Endres in [9]. Formally, in FCA, the context is a set of objects \mathcal{O} , a set of attributes \mathcal{A} , and a binary incidence relation $\mathcal{I} \in \mathcal{O} \times \mathcal{A}$ between members of \mathcal{O} and \mathcal{A} .

In our implementation, \mathcal{I} associates a word $w \in \mathcal{O}$ to the indices of its non-zero sparse coding coordinates $i \in \mathcal{A}$. FCA finds formal concepts, pairs $\langle O, A \rangle$ of object sets ($O \subseteq \mathcal{O}, A \subseteq \mathcal{A}$) such that A consists of the shared attributes of objects of in \mathcal{O} that have all the attributes in A . There is a concept ordering which forms a lattice. Following this ordering, for a query q , h will be a hypernym iff $n(q) \leq n(h)$ where $n(w)$ denotes the node in the concept lattice that introduces w .

3.3.3 Feature generation

Core feature name	
cosine	$\frac{\mathbf{q}^T \mathbf{h}}{\ \mathbf{q}\ _2 \ \mathbf{h}\ _2}$
difference	$\ \mathbf{q} - \mathbf{h}\ _2$
normRatio	$\frac{\ \mathbf{q}\ _2}{\ \mathbf{h}\ _2}$
queryBeginsWith	$Q[0] = h$
queryEndsWith	$Q[-1] = h$
hasCommonWord	$Q \cap H \neq \emptyset$
sameFirstWord	$Q[0] = H[0]$
sameLastWord	$Q[-1] = H[-1]$
logFrequencyRatio	$\log_{10} \frac{\text{count}(q)}{\text{count}(h)}$
isFrequentHypernym ³	$c \in MF_{50}(q.type)$
sameConcept	$n(h) = n(q)$
parent	$n(q) \prec n(h)$
child	$n(h) \prec n(q)$
overlappingBasis	$\phi(q) \cap \phi(h) \neq \emptyset$
sparseDifference _{$q \setminus h$}	$ \phi(q) - \phi(h) $
sparseDifference _{$h \setminus q$}	$ \phi(h) - \phi(q) $
attributePair _{ij}	$\langle i, j \rangle \in \phi(q) \times \phi(h)$

Figure 3: Features selected for classifying if pair (q, h) is a valid hyponym-hypernym pair or not. Here, q is a query vector, h is a potential hypernym’s vector, Q sequence of tokens with query phrases and H as the sequence of tokens for hypernym phrases

We defined 17 different features extracted using the sparse word vectors for the pair of expressions (q, h) consisting of query q and its potential hypernym h . Fig.3 summarizes the individual features used for classification.

3.3.4 Logistic Regression for final classification

For each appropriate (q, h) pair of words for which h is a hypernym of q , few negative samples were generated such that the training data does not include the negative ones as a valid hypernym for the given query. The valid pairs were trained using the logistic regression classifier (implementation provided in python-sklearn package was used) which was used for making predictions by determining the rankings for all possible query-hypernym candidate. From this ranking, top 15 candidates were selected and reported as the probable hypernyms for the given query.


3.4. CRIM

The team at Computer Research Institute of Montreal (CRIM) exploited combination of both “*unsupervised pattern-based approach*” and a “*supervised projection learning approach*” [6] to model the *hypernymy* relations. Officially this is the first SoTA, while unofficially this has been pushed to 2nd place by *RMM* model from Yuhang et al. [2] according to the scores evaluated in terms of *MAP*, *MRR*, and *P@5* on dataset provided in *SemEval 2018* challenge.

3.4.1 Unsupervised Pattern-Based Hypernym Discovery

The most popular approach was introduced by Hearst (1992) [14] i.e., the pattern based approach who defined special textual patterns (e.g. *Y such as X* which also called as **Hearst patterns** or Lexical-Syntactic patterns) to mine hyponym/hypernym pairs from corpora. This approach is known to *suffer from low recall* because

it assumes that hyponym/hypernym pairs will occur together in one of these patterns, which is often not the case. For instance, using the training data of *sub-task 1A*, we found that the majority of training pairs never co-occur within the same paragraph in *corpus 1A*, let alone within a pattern that suggests hypernymy. Later in our future work we'll discuss a more robust method on pattern-based hypernym discovery and how we overcome this *lower recall* problem.



Dan Jurafsky

Hearst's Patterns for extracting IS-A relations

Hearst pattern	Example occurrences
X and other Y	...temples, treasures, 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...

Figure 4: Source: *Jurafsky's Lecture on Hearst Pattern*

To solve the problem of recall, they employed *three* techniques:

1. First, identify co-hyponyms for each query and add the hypernyms discovered for these terms to those found for the query. These co-hyponyms are identified using patterns, and filtered based on distributional similarity using the embeddings (e.g., co-hyponym patterns like *X, Y, and Z*).
2. Discover additional hypernyms using a method based on most multi-word expressions as they are compositional, and the prevailing head-modifier relation is hypernymy. (e.g., for multi-word expressions such as *cold ice cream*, *ice cream* would become hypernym (since it is the headword), whereas, *cold ice cream* becomes it's hyponym.)
3. Extending the set of Hearst-like patterns which we selected empirically (e.g. *Y such as X*, *Y other than X*, *not all Y are X*, *Y including X*, *Y especially X*, *Y like X*, *Y for example X*, *Y which includes X*, *X are also Y*, *X are all Y*, *not Y so much as X*).

And they defined their algorithm as follows:

1. Create the empty set Q , which will contain an extended set of queries.
2. Search for the co-hyponym patterns in the corpus to discover co-hyponyms of q . Add these to Q and store their frequency (number of times a given co-hyponym was found using these patterns).
3. Score each co-hyponym $q_0 \in Q$ by multiplying the frequency of q_0 by the cosine similarity of the embeddings of q and q_0 . Rank the co-hyponyms in Q according to this score, keep the top n (setting $n = 5$ empirically), and discard the rest.
4. Add the original query q to Q .
5. Create the empty set of hypernyms H_q .
6. For each query $q_0 \in Q$, search for the hypernym patterns in the corpus to discover hypernyms of q_0 . Add these to H_q .
7. Add the head of each term in H_q to this set, as well as the head of the original query q .
8. Score each candidate $c \in H_q$ by multiplying its normalized frequency by the cosine similarity between the embeddings of c and q , and rank the candidates according to this score.

Note: Since this is embedding based, pattern-based co-hyponyms and hypernyms can find terms not included in the vocabulary. But those “query” which are out-of-vocabulary we simply discard them as we don't have their embeddings.

3.4.2 Supervised Projection Learning

With **projection learning** we learn a function that takes as input the word embeddings of a query q and a candidate hypernym h and outputs the likelihood that there is a hypernymy relationship between q and h . To discover hypernyms for a given query q (rather than classify a given pair of words), we apply this decision function to all candidate hypernyms, and select the most likely candidates (or all those classified as hypernyms).

This decision function can be learned in a supervised fashion using examples of pairs of words that are related by hypernymy and pairs that are not.

Supervised Model

1. Given a pair of hyponym and hypernym embeddings, (e_q, e_h) , (learned beforehand on a large unlabeled text corpus using `word2vec`), *project* the hyponym embedding using k different square projection matrices $\phi_i \in \mathbb{R}^{d \times d}$ to produce k different d dimensional vectors. Note that both e_q and $e_h \in \mathbb{R}^d$.
2. Stack those vectors vertically in row-wise fashion to obtain P matrix $\in \mathbb{R}^{k \times d}$.

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

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

3. Now check the proximity for each projections of e_q against e_h using dot product (an alias for cosine similarity).

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

4. The similarity measure, s , so obtained is fed to a *linear* layer with *sigmoid* activation function to generate logits.

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

5. To discover the hypernyms of a given query, we compute the likelihood y for all candidates and select the top-ranked ones (in our case it's 15).

Training the Model:

1. Training the model using *negative sampling*: for each positive example of a (query, hypernym) pair in the training data, we generate a fixed number, m , of negative examples by replacing the hypernym with a word randomly drawn from the vocabulary.
2. Next, we train the model to output a likelihood (y) close to 1 for positive examples and close to 0 for negative examples which is accomplished by minimizing the binary cross-entropy of the positive and negative training examples. For a particular example, this is computed as follows:

$$H(q, h, t) = t \times \log(y) + (1 - t) \times \log(1 - y)$$

where q is a query, h is a candidate hypernym, t is the target (1 for positive examples, 0 for negative), and y is the likelihood predicted by the model.

3. If we sum H for every example in the training set D (containing both the positive and negative examples), we obtain the cost function $J = \sum_{(q,h,t) \in D} H(q, h, t)$. This function is minimized by gradient descent, say using the Adam optimizer.
4. A few details of the setup we use for training are worth mentioning:
 - (a) We use a fixed number of projections (k) rather than the dynamic clustering algorithm as in Yamane et al. (2016) [19]. For our official runs, we used $k = 24$.
 - (b) The word embeddings are normalized to unit-length before training.
 - (c) For the initialization of the projection matrices, we add random noise to an identity matrix, which means that at first, the projections of a query are simply k randomly corrupted copies of the query's embedding.
 - (d) We train the model on random mini-batches containing 32 positive examples and $32 \times m$ negative examples (m being the number of negative examples which in our case is = 10).
 - (e) Dropout (= 0.5) is applied to the embeddings e_q and e_h and the query projections P . For regularization, we also use *gradient clipping*, with a clip of $1e^{-4}$, as well as *early stopping*, with patience value = 200.
 - (f) We sample positive examples using a function based on the frequency of the hypernyms in the training data, such that we subsample (q, h) pairs where h occurs often in the training data. The probability of sampling (q, h) is given by:

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

where $\text{freq}(h)$ returns the frequency of h in the training data.

- (g) The word embeddings are optimized (or “fine-tuned”) during training.
 - (h) We use a multi-task learning setup whereby we train two separate logistic regression classifiers, each with their own parameters W and b , and use one for queries that are *named entities*, and the other for queries that are *concepts*. The rest of the parameters (i.e. the projection matrices ϕ) are shared.
 - (i) **Extras:** The max epoch is set to 1000. We use the **Adam** optimizer with $\beta_1 = \beta_2 = 0.9$ and with a **learning rate** of $2e^{-4}$ for all datasets. We choose two separate embedding transformation matrices for two different query types. When initializing these projection matrices in the mapping function, we add random noises of *Gaussian distribution* (zero mean and $\frac{1}{200}$ variance) to an identity matrix. Finally, we implement our model using **PyTorch** on a **Linux** machine with a GPU device **Nvidia RTX 2080 12GB**. More info can be found in **hparams.conf** file in code.
- (**Note** that the various hyperparameters mentioned above were tuned on the *trial set* (i.e. development set) provided for all the sub-tasks.)

3.4.3 Hybridization: How Supervised and Unsupervised are fused?

1. Select top 100 candidates according to each hyponym, normalize their scores and sum them.
2. Rerank the candidates according to this new score. This reranking function favours candidates found by both systems, but also gives a chance to strong candidates found by a single system.

With that being said, our main goal is to design model that mostly motivated from **CRIM** and on top of that w.r.t projection matrix what all alternatives we can look upon is what remaining to get answered in our future work.

4. Experiments and Results

We evaluate the performance of our model on SemEval-2018 Task 9 [7] benchmark for hypernym discovery. This shared task consists of five different subtasks covering both general-purpose (multiple languages-English, Italian, and Spanish) and domain-specific (Music and Medicine domains) tasks. For each subtask, a large textual corpus, a vocabulary including all valid hypernyms and a training and testing set of hyponyms and its gold hypernyms are provided. But for subtask **1B** (the Italian corpus from Baroni et al. 2014 [3]) and **1C** (the Italian corpus from Cardellino 2016 [8]), the corpus’s google-drive link has been made dormant. The same goes for subtask **1A** (the english corpus from UMBC (Han et al. 2013) [12]), but here we download the super-corpus available on this [site](#) and pre-processed it to make our version of english corpus in `.txt` format that takes up a space of **18GB**. To have embeddings for subtask 1B and 1C, we referred to some open source embeddings files (spanish¹² and italian³⁴). The summarized statistics of the datasets are shown in Table 2. For more details, we refer the reader to the original SemEval-2018 Task 9⁵ (Camacho-Collados et al., 2018) paper.

While the subtask 2A (medical) [15] and 2B (music) [18] have their corpus available on site of SemEval-2018 Task 9⁵

subtask	corpus size	#train	#trial (dev)	#test
1A (English)	18G	1500	50	1500
1B (Italian)	–	1000	25	1000
1C (Spanish)	–	1000	25	1000
2A (Medical)	800M	500	15	500
2B (Music)	500M	500	15	500

Table 2: Data set statistics

Three metrics were used for the performance evaluation.

- Mean Average Precision (**MAP**): For a given query word, average precision(AP) is the average of the correctness of each obtained hypernym from the search space. MAP is the mean of this value among all queries in the data set.

¹Zoo for several language embeddings. Took spanish-eswiki from [here](#).

²Took another Spanish word2vec embeddings (SBW-vectors-300-min5.txt) trained on *Spanish Billion Word Corpus (SBWC)* [8] from [here](#).

³Italian-300-dimension embedding from this [site](#)

⁴Italian-128-dimension embedding from [Italian NLP Lab](#) on itWac [3] corpus. [Link](#)

⁵<https://competitions.codalab.org/competitions/17119>

- Mean Reciprocal Rank (**MRR**): Since MAP ignores the exact rank of the true hypernyms, we introduce the Mean Reciprocal Rank (MRR) metric which focuses on the top results performance. It is the average of the reciprocal ranks over all queries. The reciprocal rank of an individual query is the reciprocal of the rank in which the first true hypernym is returned.
- Precision at K (**P@K**): Precision at K is the proportion of the top-K results that are true hypernyms of a given query.

4.1. Learning Word Embeddings

We learned term embeddings of **300** dimensional for all queries and candidates using the pre-tokenized corpora provided for sub-tasks 1A, 2A, and 2B via the standard **skip-gram word2vec** algorithm with negative sampling (Mikolov et al., 2013) [16]. We preprocessed the corpora by converting all characters to lower case and replacing multi-word terms found in the vocabulary (candidates and lower-cased queries) with a single token, starting with trigrams, then bigrams.

4.2. Results & Plots

A summary of our system’s results is shown in Table 4. This table shows the mean average precision (MAP), mean reciprocal rank (MRR) and precision at rank 5 (P@5) for all of our systems.

Model	General - English (1A)			Domain - Medical (2A)			Domain - Music (2B)		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
Baseline	0.41	0.41	0.16	6.63	6.68	1.76	12.94	13.03	3.99
TaxoEmbed	9.75	9.67	2.39	18.57	19.21	13.16	18.29	18.31	12.68
FCA	12.14	26.40	11.43	33.72	50.83	32.14	33.62	51.12	32.02
CRIM	23.53	34.44	20.43	33.80	42.61	35.16	46.20	62.39	47.85

Table 3: Performance comparison on different models on the benchmark datasets.

Model	1A			1B			1C			2A			2B		
	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5	MAP	MRR	P@5
CRIM	23.53	34.44	20.43	21.37	34.29	17.44	33.41	52.23	29.23	33.80	42.61	35.16	46.20	62.39	47.85

Table 4: CRIM Scores on All Subtasks.

	1A (without embedding normalization)	1A (with Batch Size = 64)	1B (Pre-trained embedding from itWac corpus with 128 as embedding size)	1C (Spanish Embeddings from SBWC Corpus)	2B (without embedding 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

Table 5: With different pre-trained embedding file, different settings of hyper-parameters, or different corpus, we obtained the above scores (tested only on CRIM)

Query	Predictions (Hypernyms)
Suzy Favor Hamilton wicketkeeper aquamarine tenpence	athlete , sportsperson , person , competitor, sport, olympic sport,... cricketer , sportsperson , athlete , competitor , footballer, person ,... stone , crystal, precious stone, pebble , gem, rock, gemstone,... monetary unit, metal money , note of hand , person, silver coin, coin,...
vegetarian Local Group Swarthmore hypostasis	dessert, dish, recipe, veggie, food product, organic food, meal, salad,... voluntary association, locale, coalition, club, country, mapmaking,... university, college, educational institution, school, student,... figure of speech, intellection, philosophy, ordinary language,...

Table 6: Examples of predictions made by our system on the test queries of 1A. Correct predictions are in bold. Midline separates high-accuracy examples from low-accuracy examples.

4.2.1 Plots

For **CRIM**, we plot the profile of *BCE* loss and *MAP* score against epochs for training of its supervised module. We present 4 plots for each subtask that shows the epoch-wise variation of:

- Training Loss on Positive Pairs.
- Training Loss on Negative Pairs.
- Dev Loss.
- MAP score.

1A: English

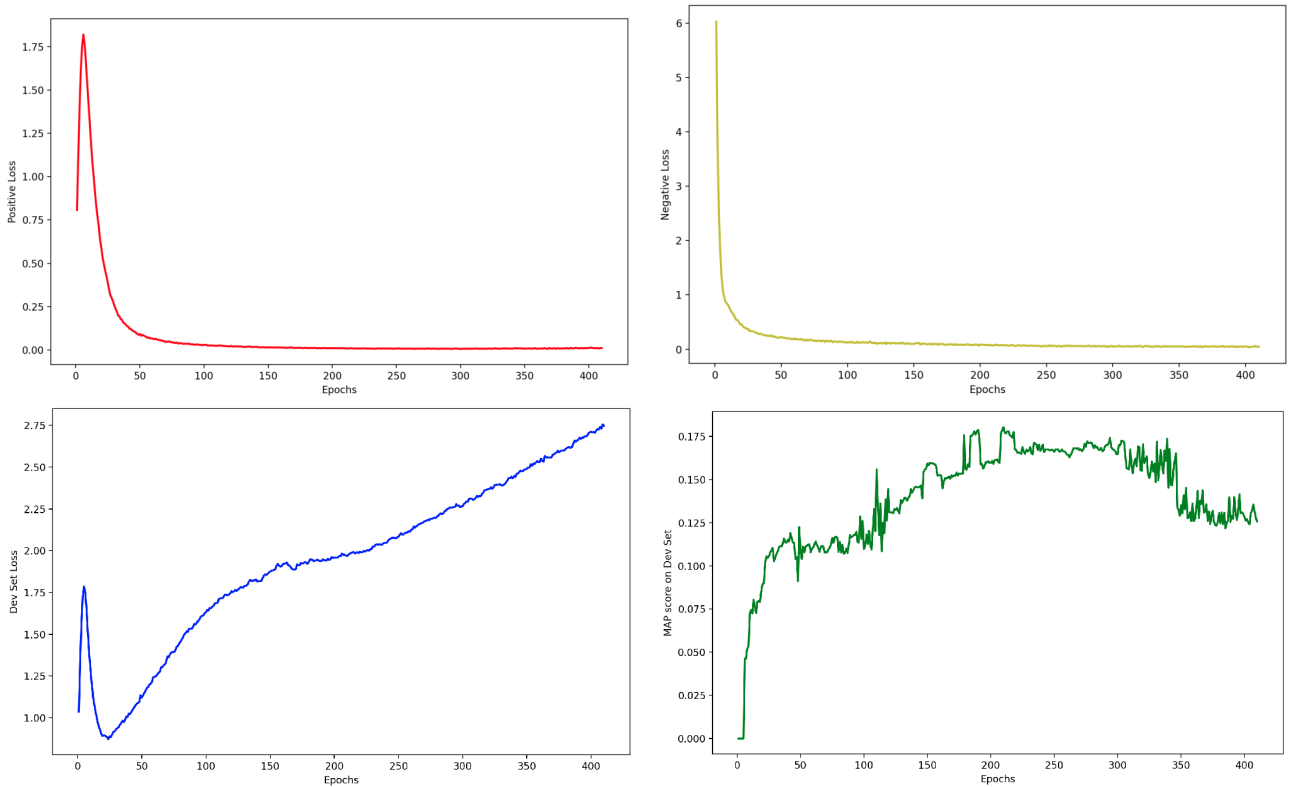


Figure 5: Plots for English (1A) subtask

1B: Italian

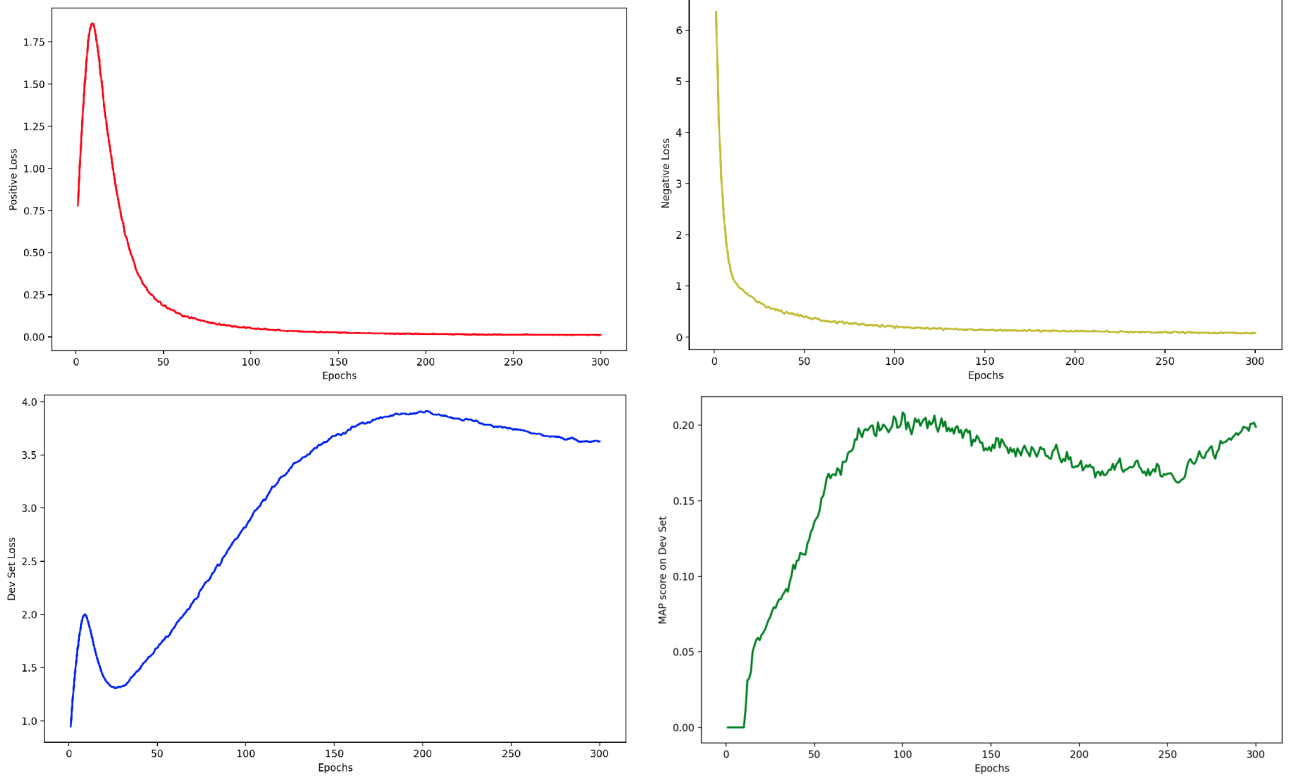


Figure 6: Plots for Italian (1B) subtask

1C: Spanish

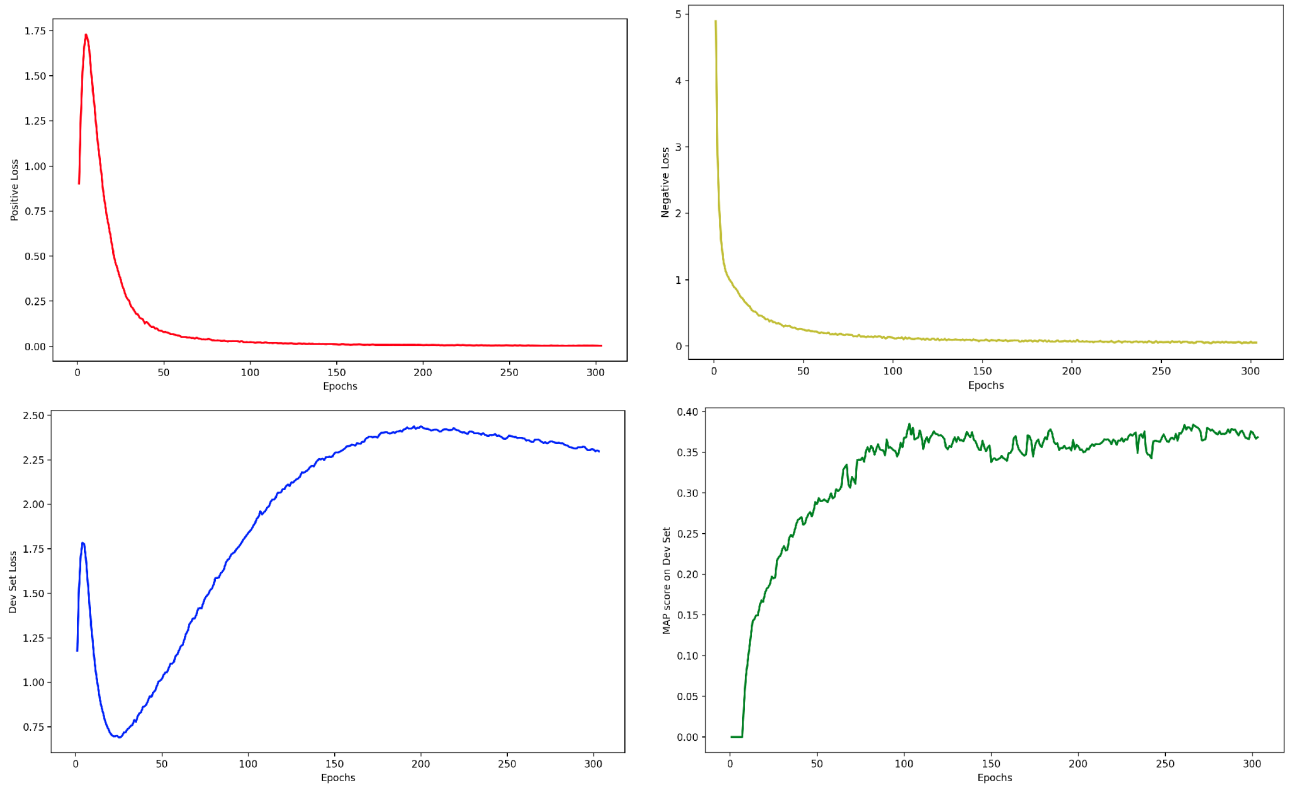


Figure 7: Plots for Spanish (1C) subtask

2A: Medical

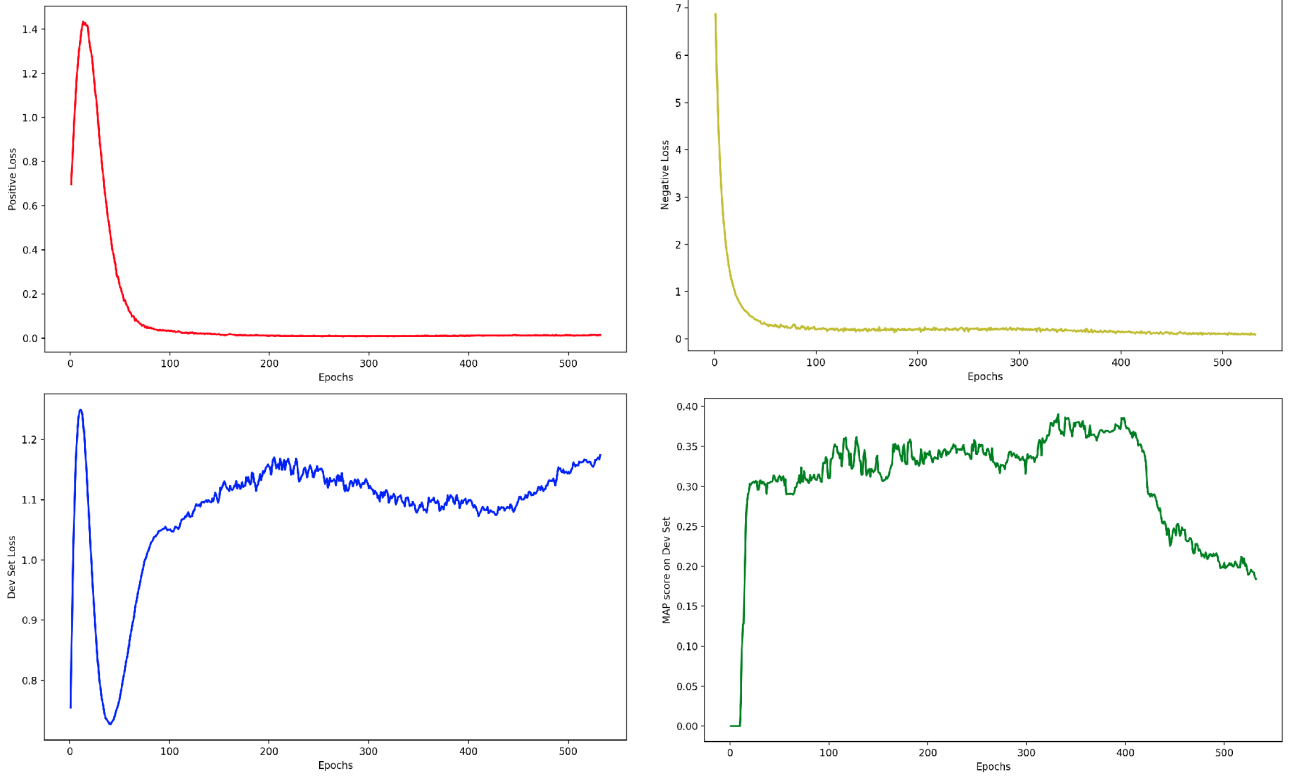


Figure 8: Plots for Medical (2A) subtask

2A: Music

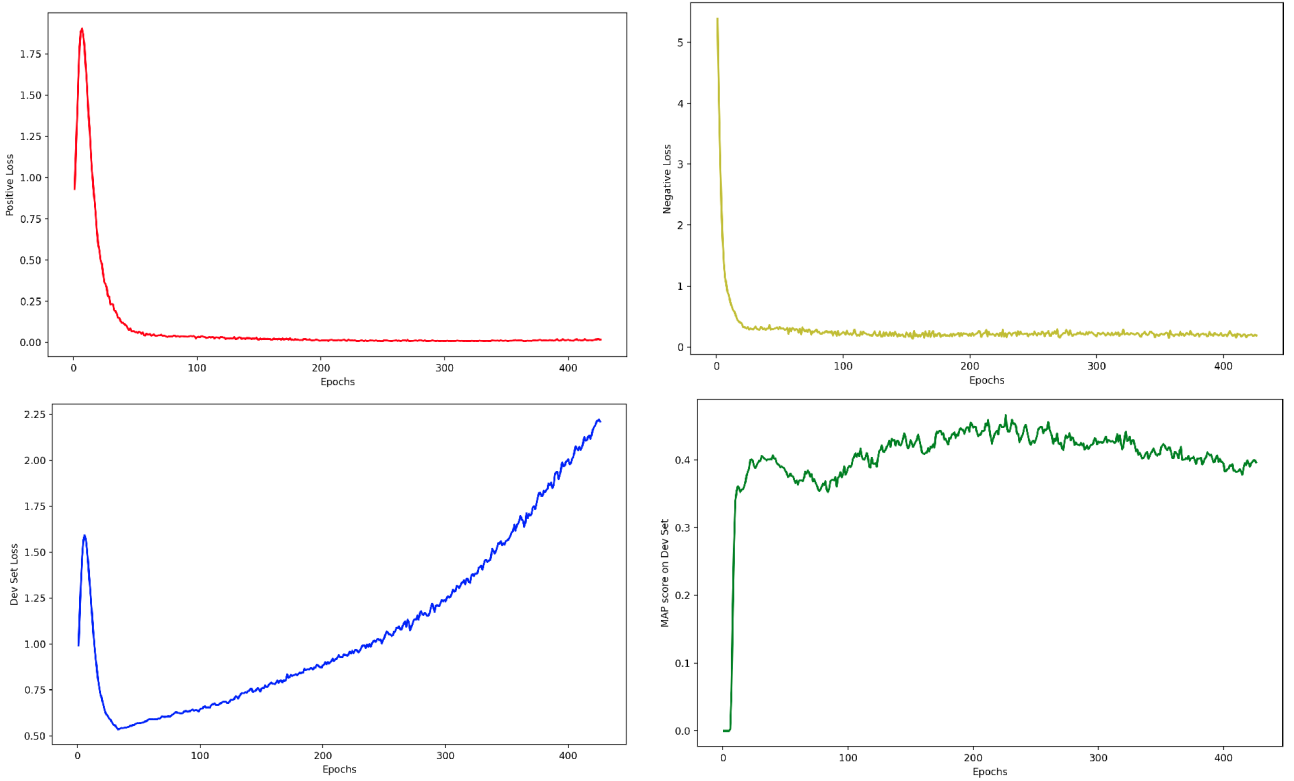


Figure 9: Plots for Music (2B) subtask

5. 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. Hypernym discovery is a basic task in natural language processing. Existing studies focus on designing better models for discovering better mapping functions from hyponyms to hypernyms. Our approach to hypernym discovery combines a novel supervised projection learning algorithm and an unsupervised pattern-based algorithm which exploits co-hyponyms in its search for hypernyms. This hybrid approach produced very good results on the hypernym discovery task, and was ranked first on all 3 sub-tasks for which we submitted results. However, most of these hybrid models are two separate processes and the supervised part highly depends on the pre-defined “is A” patterns. To build a uniform hybrid model still remains an open problem. We will study these open problems in our future work.

References

- [1] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.
- [2] Yuhang Bai, Richong Zhang, Fanshuang Kong, Junfan Chen, and Yongyi Mao. Hypernym discovery via a recurrent mapping model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2912–2921, Online, August 2021. Association for Computational Linguistics.
- [3] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don’t count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [4] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10, 2011.
- [5] Gábor Berend. Sparsity makes sense: Word sense disambiguation using sparse contextualized word representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8498–8508, Online, November 2020. Association for Computational Linguistics.
- [6] Gabriel Bernier-Colborne and Caroline Barrière. CRIM at SemEval-2018 task 9: A hybrid approach to hypernym discovery. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 725–731, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.
- [7] Jose Camacho-Collados, Claudio Delli Bovi, Luis Espinosa-Anke, Sergio Oramas, Tommaso Pasini, Enrico Santus, Vered Shwartz, Roberto Navigli, and Horacio Saggion. Semeval-2018 task 9: Hypernym discovery. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018); 2018 Jun 5-6; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 712–24*. ACL (Association for Computational Linguistics), 2018.
- [8] Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, August 2019.
- [9] Dominik Endres, Ruth Adam, Martin A. Giese, and Uta Noppeney. Understanding the semantic structure of human fmri brain recordings with formal concept analysis. In Florent Domenach, Dmitry I. Ignatov, and Jonas Poelmans, editors, *Formal Concept Analysis*, pages 96–111, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [10] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado, May–June 2015. Association for Computational Linguistics.
- [11] Alona Fyshe, Leila Wehbe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. A compositional and interpretable semantic space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 32–41, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

- [12] Lushan Han, James Mayfield Abhay L. Kashyap Tim Finin, and Johnathan Weese. Umhc-ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, June 2013.
- [13] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992.
- [14] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, 1992.
- [15] PubMed MEDLINE.
- [16] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.
- [17] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.
- [18] Espinosa-Anke L. Oramas, S., Saggion H. Sordo M., and Serra X. Elmd: Entity linking for the music domain dataset, 2016.
- [19] Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1871–1879, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.