NounAtlas: A Dual Network Approach for event synsets Identification and Classification within VerbAtlas Frames

Cristiano Bellucci, Omar Bayoumi
Sapienza University of Rome
Matricola: 1760390, 1747042
bellucci.1760390@studenti.uniroma1.it
bayoumi.1747042@studenti.uniroma1.it

1 Abstract

The goal of this project is to create a nominal resource corresponding to and integrated with **VerbAtlas** (Di Fabio et al., 2019). To achieve this, the derivational related nominal form of verbs associated with **VerbAtlas** was employed, and the hypernym tree was traced back to the **Event synsets**. While it is certain that a nominal synset that reaches **Event** is an event, the opposite is not always true. In addition, a nominal synset can be associated with **multiple VerbAtlas frames**, resulting in **ambiguity**. In order to overcome these issues, two neural networks were created: one for **identifying** Event synsets (2) and another for **disambiguating** nominal synsets (3). This report will explore the methodology and results of the project.

2 Event Identification

In this section, the results and methods used for the process of identifying event synsets will be analyzed.

2.1 Dataset Creation and Preprocessing

Two labels are required for the construction of this dataset: *event* and *not_event*. All synsets generated from the hyponym tree of **Event** synsets are considered events. The not events were chosen from synsets as general as possible (Table 1) through a **manual analysis** of their **WordNet** (Miller, 1995) definitions. From these synsets, their hyponyms trees are generated. All these synsets combined with the **most general** possible synsets constitute the **not event synsets**. However, the number of not event synsets is much **larger** than the number of event synsets, and to have a **balanced** dataset it is necessary to **reduce** the number of not event synsets by **maintaining the original proportions** (Table 1). The dataset is then split into: **70% training set**, **15% dev set**, **15% test set**. The choice to maintain the proportions was made because synsets belonging to the **same tree** express entities of the **same type**, although having different definitions. A **random choice** among the not event synsets could have fallen on many **synsets belonging to the same tree**, thus making the **dataset biased** on this type of entity. Finally, a *"final set"* was created containing all those synsets associated with **VerbAtlas** that did not reach **Event synsets** and are not present in the other sets.

Each **sample** in the dataset consists of the **synset definition**, **its direct hypernyms**, if any, and **POS tagging of the synset definition**.

2.2 Architecture

The model architecture (Figure 1) uses a transformer, specifically **bert-base-uncased**, whose output is subsequently fed into a **BLSTM**. In order to enhance the model's performance, an **attention mask** is incorporated following the **BLSTM**. Finally, a **linear binary classifier** with a **sigmoid activation function** is used to provide the predictions in the output. The inclusion of the attention mask is intended to add relevant information and improve the results.

2.3 Training

During the training phase of this study, the model was trained for a maximum of **100 epochs**. A **learning rate of 4e-4** was chosen, because a learning rate that is too high can cause the model to overfit the training data quickly, while a learning rate that is too low can lead to slow convergence. The chosen learning rate was likely determined through a hyperparameter search that involved testing different values. This learning rate yielded the best results for the model.

To further reduce overfitting, **weight decay** was applied to the **BLSTM**. Weight decay involves adding a penalty term to the neural network's cost function, resulting in a decrease in weights during backpropagation. Additionally, an **early stopping**

algorithm was implemented, which counted the number of epochs where the current validation loss was greater than the validation loss obtained in at least half of a time window of eight epochs before. If the counter reached five, training was stopped.

Adam was chosen as the optimizer because it produced better results in less time than **SGD**. A **binary cross-entropy loss function** was used since this is a binary classification task.

2.4 Results

Numerous tests were conducted by varying the values of the parameters (refer to Figure 5). The best results were obtained with the following hyperparameters: **Lr 4e-4**; **Wd 0.01**; **Transformer Lr 5e-5**; **Batch size 32**; **Hidden dim LSTM 200**; **POS Embedding Dim 128**; **Num Layer LSTM 3**; **Trasnformer dropout 0.2**; **Dropout 0.5**; **All hypernyms True**.

The other hyperparameters and the results obtained are given in Table 2 and Table 3. The **"All hypernyms"** hyperparameter indicates whether all definitions associated with all hypernyms in the input synset were added in the construction of the input. **Better results** were obtained using the **"All hypernyms"** hyperparameter. This is because the definitions of some synsets could be misleading to the network. In particular, it was noted that the synsets related to **emotions** very often were confused by the network as events because of the synset definition itself. In these cases, the choice of **adding hypernames** helped a lot because it allowed the network to have **additional information** about the meaning of the input synset and thus understand that it was not an event. The use of weight decay, as can also be seen in Table 3, improved the results because the network tended to overfit very quickly not reaching optimal values of validation loss. The results of the **F1 score**, **accuracy**, **precision**, and **recall** are shown in Table 3. As can be seen, the results are very high because the task is simple, so the network learns well to distinguish between events and not events associated with the **training**, **dev**, and **test** datasets. However, from the tests performed on the *"final set"*, it appears that the classification of events/not events seems to be **less efficient**. This is probably because the training, dev, and test samples were **generated** from the **"Event" synsets** and **general not event synsets** (2.1), while there may be **unrelated** synsets in the *"final set"*. Consequently, these scores should be taken as an **indicative** value and not as actual network efficiency. A manual analysis was performed to assess the actual efficiency of the model.

2.5 Final Results

The best model was used to rank the elements of the *"final set"*. If the network prediction for the event label was greater than **0.99**, then that synset was considered an **event**. It was decided to use **0.99** as the threshold instead of **0.5** because we wanted to be sure that the synsets were **truly** event synsets. This threshold value was chosen after careful manual evaluations. In fact, some synsets with scores between **[0.5, 0.99)** do **not seem to be events**, for example, the synset *"conclusion.n.08"* with the definition "the last section of a communication" has a prediction of **0.89**, but it does not seem to be an event. In most cases, the network correctly predicts events using this threshold. However, in a few cases, it is wrong, for example, the synset *"green_light.n.01"* with the definition "a signal to proceed". A few times the network also predicts **events as not events**. However, this does **not affect** the performance of the network, because in the next step, **only events** are used. Consequently, fewer samples will be added to the next step dataset (3.1) because of these network errors. At the end of the process, **854 new event synsets** and **1536 not event synsets** were obtained.

3 Event Classification

This section will review the results and methods used for the process of **disambiguation** of event synsets associated with multiple **VerbAtlas** frames.

3.1 Dataset Creation and Preprocessing

The first step in building the dataset is to **find** all the event synsets. New event synsets obtained in the **identification step** are added to these synsets. The synsets obtained so far can be **ambiguous** or **unambiguous**. Each sample consists of a **synset definition**, **associated POS tagging**, and a **computation done with the VerbAtlas API**. This computation consists of associating with each word in the definition the symbol *"~"* if the word is **not a verb**, otherwise the **frame it belongs to**. The idea behind this choice is to use the information in the synset definition, in terms of the **VerbAtlas** frame, as a **hint** for classifying the synset into the **correct** frame.

It was decided to divide the dataset into **80%**

training, **10% development**, and **10% test** sets. All of these sets are composed of **unambiguous** samples. These percentages were selected to assign greater importance to the **training phase**, considering the **limited number of samples** available. The *"final set"* consists of all **ambiguous** samples. Various approaches were tried for constructing the training, development, and test datasets.

3.1.1 First Approach

The first approach to dataset construction was to analyze the **unambiguous synsets** and determine which frame they were associated with. In this way it was possible to obtain the **frequency** of the various **VerbAtlas frames** (Figure 2). Frames, which represent the **labels** of the input samples, that had a **frequency below a certain value were discarded**, which consequently **reduced** the size of the dataset on which the network was trained. In fact, this method resulted in the discarding of about **half** of the **unambiguous synsets**. For **each remaining frame** after skimming, the **80% 10% 10%** split was done to keep the dataset as **balanced** as possible.

3.1.2 Second Approach

Due to the inadequacy of the previous method, which yielded very low scores primarily due to the limited size of the dataset, it was deemed necessary to **modify the dataset construction approach**.

In this alternative approach, a new label called *"other"* was introduced, encompassing all frames with a **frequency below a specific threshold but greater than 2**. The chosen threshold value was motivated by the requirement of having at least one sample for each set (train, dev, test), **necessitating a frame frequency of at least 3**. After that, for each frame in *"other"*, the **80% 10% 10%** split was done to keep the dataset as **balanced** as possible. Nonetheless, despite this adjustment, **numerous samples were still discarded**, prompting further **modifications** to the *"final set"*.

Consequently, if a sample contained **one or more labels** that were **not present** in the *"other"* category, it was **eliminated**. This step was **essential** since the model lacked the necessary information to accurately classify the sample within the *"other"* class, as it was **not trained to predict** the *"other"* label for these specific frames. Additionally, **any sample with two or more labels** assigned to the *"other"* category was also **discarded**. This decision was made due to insufficient information to properly disambiguate the sample if the model predicted *"other"* as the label.

3.1.3 Final Approach

To address the challenge depicted in Figure 3, where there are **numerous frames** with a **frequency less than 3**, it becomes crucial to **avoid discarding** any data in order to enhance performance. Consequently, a further modification was introduced to tackle this issue.

In this updated approach, **frames** with a **frequency below 3** are assigned to the label *"other"*. In particular, they are **grouped** into a single subset called *"few_occurencies"*. Subsequently, this subset is randomly partitioned into **train**, **dev**, and **test** sets, maintaining the proportions of **80%**, **10%**, and **10%**. By adopting this methodology, **no data is discarded**.

This revised strategy also partially resolves the problem encountered in the *"final set"*, eliminating scenarios where a label is **not** in *"other"*.

3.2 Architecture

The architecture used (Figure 4) for the classification phase is similar to the one used for the identification phase. However, certain variations exist between the two. In the classification phase, an **extra embedding** associated with the **VerbAtlas** API call is incorporated into the input. As a result, the input provided to the **BLSTM** takes the following form: **output transformer + POS embedding + VerbAtlas frame embedding**. Furthermore, the **binary classifier with sigmoid** is substituted with a **multiclass classifier employing log softmax**.

In terms of labeling, the classification phase incorporates the labels obtained during **dataset creation**, plus *"other"* (second and final approaches).

3.3 Training

During this phase, the model was trained for a maximum of **100 epochs**, and **early stopping** was also implemented. However, unlike the identification phase, **weight decay was not utilized** in this stage as it yielded inferior results. Given that the task involved multiclass classification, the **cross entropy loss function** was employed.

Given the **dataset's inherent class imbalance** (Figure 2), various weightings were explored and applied to the loss function, resulting in different outcomes. These weightings included: **No specific weights assigned**; **An average weighting derived from a combination of Gaussian and uniform**

distributions; **Implementation of the log softmax approach**.

Among the available optimizers, **Adam** was selected as the preferred choice.

3.4 Results

Numerous tests were conducted by varying the values of the parameters (refer to Figure 6). The best results were obtained with the following hyperparameters: **Lr 5e-5**; **Transformer Lr 5e-5**; **Batch size 16**; **Hidden dim LSTM 200**; **POS Embedding Dim 128**; **Num Layer LSTM 2**; **Trasnformer dropout 0.2**; **Dropout 0.3**; **Weight type Log softmax**; **Num label occurrences 11**.

The other hyperparameters and the results obtained are given in Table 4 and Table 5. The **"weights type"** variable denotes the specific weightings applied to the loss function, as illustrated in the subsection 3.3 and displayed in Table 4. The results indicate that **not using any weights** resulted in **lower performance** due to the **dataset's inherent class imbalance**. Slightly inferior outcomes were observed when utilizing a **combination of uniform and Gaussian distributions** as weights. The idea behind this approach was to incorporate weights derived from classes with similar frequencies, as represented by the Gaussian distribution. However, employing this technique led to **poorer results** since f**rames with similar frequencies lacked meaningful correlations**, thereby hindering effective dataset balancing.

The best results were achieved by utilizing the log softmax approach. Unlike other weight types used, **log softmax maintained frame independence**. It is important to note that the log softmax employed in this study was **slightly modified**, wherein **positive values were applied and then reversed**. This adjustment ensured that **frames with lower frequencies were assigned higher weights**.

Additionally, an experiment was conducted using the **"bert-large"** transformer model. However, this particular model yielded **disappointing results**, potentially attributed to the **limited dataset size**, which hindered proper generalization and fine-tuning of such a large model.

The hyperparameter **"Num label occurrences"** determined the **threshold** below which frames were **discarded in the first approach**, **assigned to the "other" class in the second approach**, **and assigned to the** *"other"* + *"few occurrences"* **class in the final approach**.

The final model produced **F1 scores**, **precision**, **accuracy**, and **recall** results approximately around **53%**. Although weights were employed, the **overall results were not highly satisfactory**. This may be attributed to the significant number of frames with **low frequencies**, posing challenges for the network in accurately classifying synsets into their respective frames.

3.5 Final Results

The optimal model was employed to disambiguate the synsets within the *"final set"*. **Disambiguation** was accomplished by examining the **distribution** resulting from the model's predictions, specifically focusing on the values corresponding to the classes associated with the **ambiguous synsets**. If **any** of these frames were present within the *"other"* set, the value associated with the *"other"* class was **extracted** from the prediction. The **disambiguation process** is done by choosing the class with the **highest probability**. If *"other"* was the outcome of disambiguation, the synset was **disambiguated using the original class** that had been present in the *"other"* set (at most one class, subsubsection 3.1.2).

However, there were **two scenarios** in which synsets were deliberately kept ambiguous. The first case occurred when **at least one** of the classes associated with the ambiguous synset was **not included** in the classes used during the **training phase**. This was due to the lack of samples necessary to determine whether the absent class was the correct one for classification or not.

The second case arose when the highest predicted value among the selected classes **fell below a specific threshold (0.002)**. This decision was made to ensure that results below the threshold were deemed **unreliable**, and therefore, **maintaining ambiguity** was preferred over **potentially misleading classifications**.

Out of the analyzed synsets, **743 were disambiguated**, while **72 were intentionally left ambiguous**. Among the ambiguous synsets, **20 had a class that was not present in the training data**, while **52 were not above the threshold**.

3.6 Conclusions and future improvements

The process of identification and classification presented **notable challenges**, particularly in the *"final set"* where certain samples were incorrectly predicted due to **ambiguous definitions**. The classification step proved to be even **more intricate** as compromises had to be made to disambiguate

synsets. The **limited size** of the dataset posed a significant hurdle, preventing the network from obtaining sufficient information to accurately classify synsets into frames. This problem surely is due to the fact that **most synsets** appear with a **small frequency** and consequently, despite the use of weights, the network has difficulty learning how to classify a synset in these frames. These factors contribute to the overall poor performance of the network. A **larger dataset** would undoubtedly enhance the results by providing a **higher frequency of frames**. Furthermore, certain frames were entirely **discarded** as they lacked any associated synsets. Consequently, if the network attempts to predict synsets to be classified in these frames, it is bound to make errors.

Manual analysis of the results indicated instances where certain synsets could be classified into **multiple frames**, which can be attributed to the **conceptual similarity** of these frames. For this reason, **merging them into a single frame** could potentially improve network performance. This consolidation would **increase** the overall frequency of frames, thereby **minimizing** the likelihood of **discarding** frames or assigning them to the *"other"* class. If the network were to achieve improved results, it could serve as a valuable **VerbAtlas frame embedding**, **enhancing the understanding of sentence meaning** and facilitating a deeper comprehension of the expressed event.

Synsets	Number of hyponyms	Proportions
thing.n.08	6	-
object.n.01	25695	53.57%
set.n.02	26	-
substance.n.04	1	-
matter.n.03	6115	12.75%
otherworld.n.01	1	-
measure.n.02	2122	4.42
group.n.01	7916	16.50%
attribute.n.02	6120	12.76%

Table 1: List of the most general synsets with the number of hyponyms and the proportions used to balance the dataset. The "-" means that all the hyponyms are taken

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
Lr	4e-4	5e-4	4e-5	4e-4	4e-5	4e-5	4e-4	4e-4
Wd	0.01	0.01	0	0.01	0	0	0.01	0.01
Transformer lr	5e-5	5e-5	5e-6	5e-5	5e-6	5e-6	5e-5	5e-5
Batch size	32	32	16	32	16	16	32	32
Hidden dim LSTM	200	200	300	200	200	300	200	200
POS Embedding dim	128	128	256	128	256	256	128	128
Num layer LSTM	3	2	3	3	2	3	2	2
Transformer Dropout	0.2	0.2	0.5	0.2	0.5	0.5	0.3	0.3
Dropout	0.5	0.5	0.3	0.5	0.3	0.3	0.6	0.6
All Hypernyms	True	False	False	True	True	True	False	True

Table 2: Identification step models and their hyperparameters

	F1	Accuracy	Precision	Recall
Model 1	**0.9924**	**0.9924**	**0.9924**	**0.9924**
Model 2	0.9912	0.9912	0.9912	0.9912
Model 3	**0.9550**	**0.9550**	**0.9550**	**0.9550**
Model 4	0.9726	0.9726	0.9737	0.9726
Model 5	0.9912	0.9912	0.9912	0.9912
Model 6	0.9886	0.9886	0.9887	0.9886
Model 7	0.9920	0.9920	0.9920	0.9920
Model 8	0.9903	0.9903	0.9903	0.9903

Table 3: Identification step model results

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9	Model 10	Model 11
Lr	5e-5	5e-4	5e-4	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
Wd	0	0	0	0.002	0.001	0.001	0	0.001	0.001	0.001	0.001
Transformer lr	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	4e-5	5e-5	5e-5	5e-5	5e-5
Batch size	16										
Hidden dim LSTM	200										
POS Embedding dim	128										
Frame Embedding dim	128										
Num layer LSTM	2	2	2	2	3	2	2	2	2	2	2
Transformer Dropout	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Dropout	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.4	0.4	0.4	0.4
Num label occurrences	11	10	12	11	11	11	11	11	4	4	4
Weights Type	log softmax	log softmax	log softmax	log softmax	log softmax	log softmax	log softmax	log softmax	log softmax	no weights	uniform + gaussian
Transformer	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-base-uncased	bert-large	bert-base-uncased	bert-base-uncased

Table 4: Identification step models and their hyperparameters

	F1	Accuracy	Precision	Recall
Model 1	**0.5216**	**0.5255**	**0.5507**	**0.5255**
Model 2	0.4111	0.4124	0.4651	0.4124
Model 3	0.4316	0.4102	0.5472	0.4102
Model 4	0.5084	0.5122	0.5388	0.5122
Model 5	0.4928	0.4878	0.5355	0.4878
Model 6	0.5072	0.5078	0.5442	0.5078
Model 7	0.4913	0.5055	0.4981	0.5055
Model 8	0.4944	0.5055	0.5213	0.5055
Model 9	**0.0003**	**0.0118**	**0.0001**	**0.0118**
Model 10	0.3209	0.3628	**0.3160**	0.3628
Model 11	**0.3196**	**0.3540**	0.3233	**0.3540**

Table 5: Classification step model results. Green and red represent the best and worst results, respectively. Blue is the model using "bert-large" whose results are not comparable to those of the other models

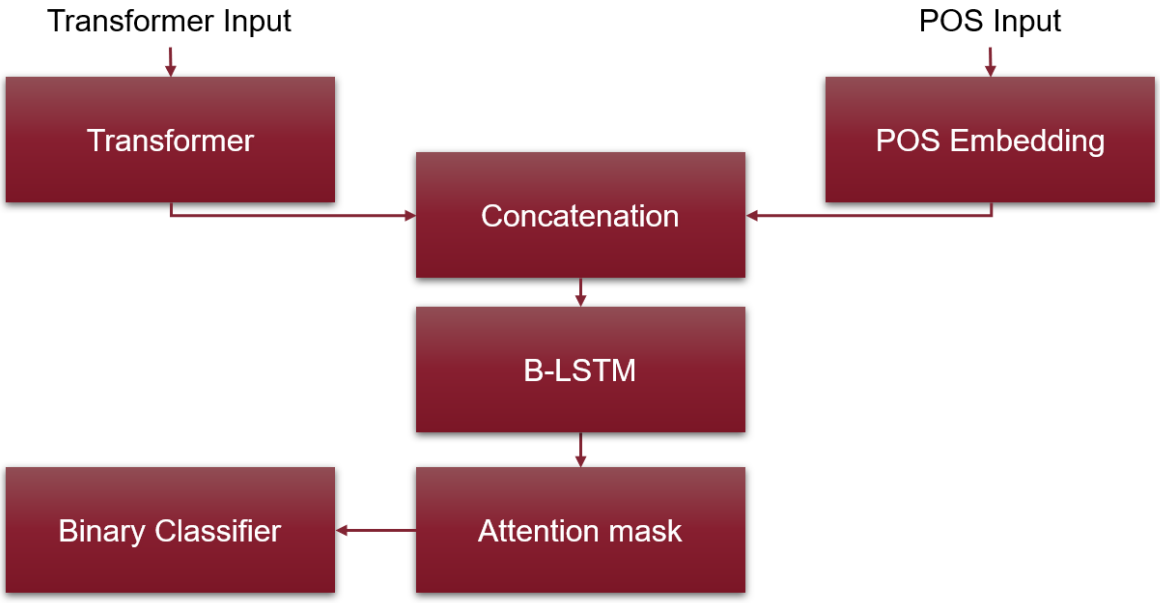


Figure 1: Event Identification model architecture.

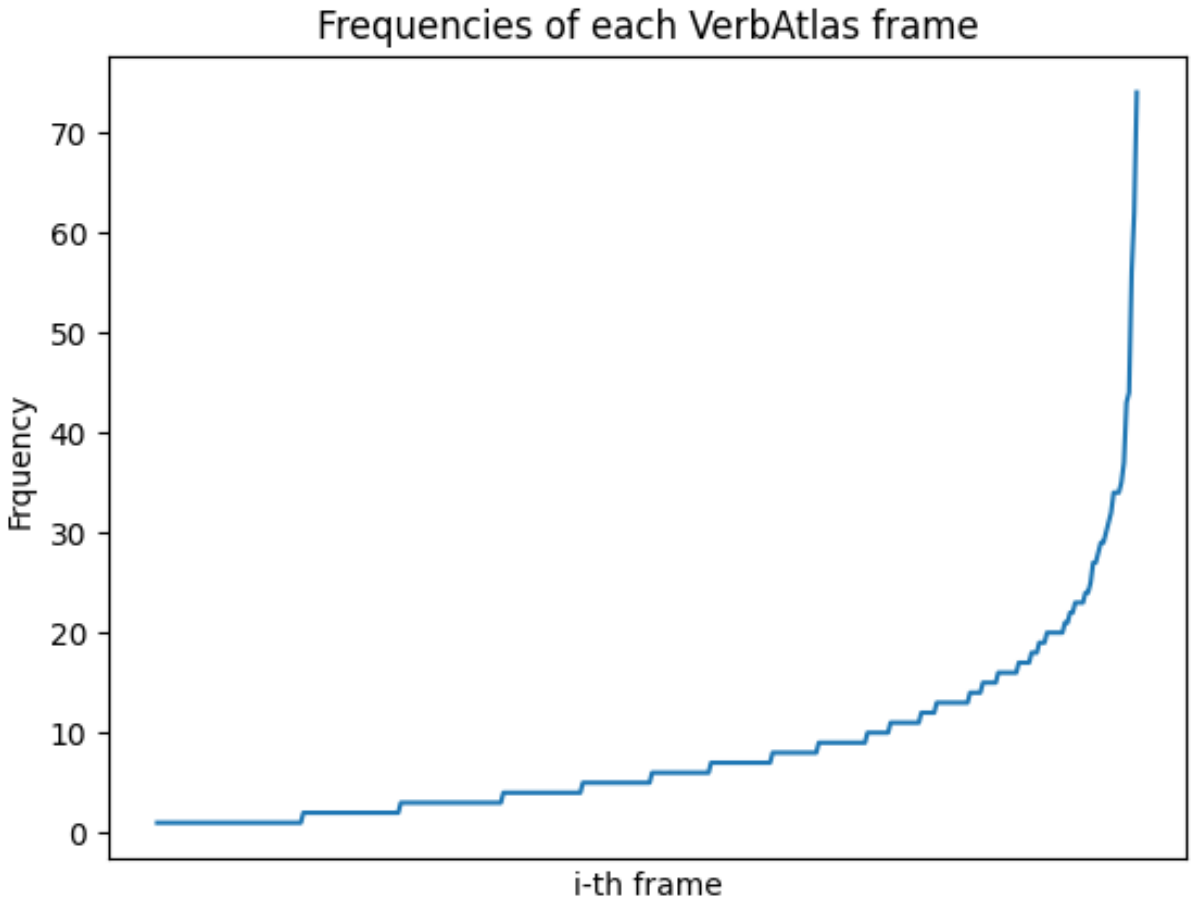


Figure 2: Frequiency of VerbAtlas frames.

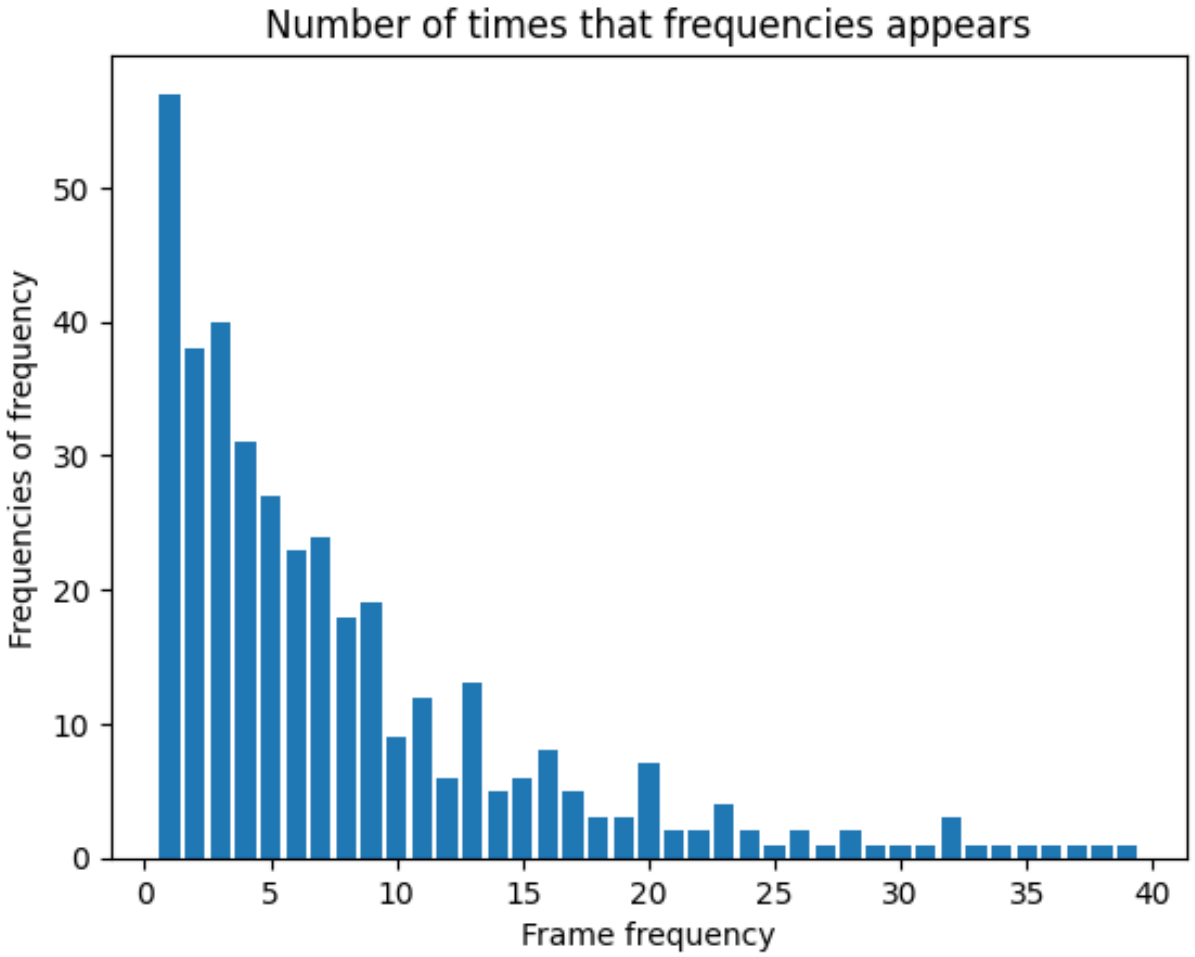


Figure 3: Frequency of the VerbAtlas frames frequencies.

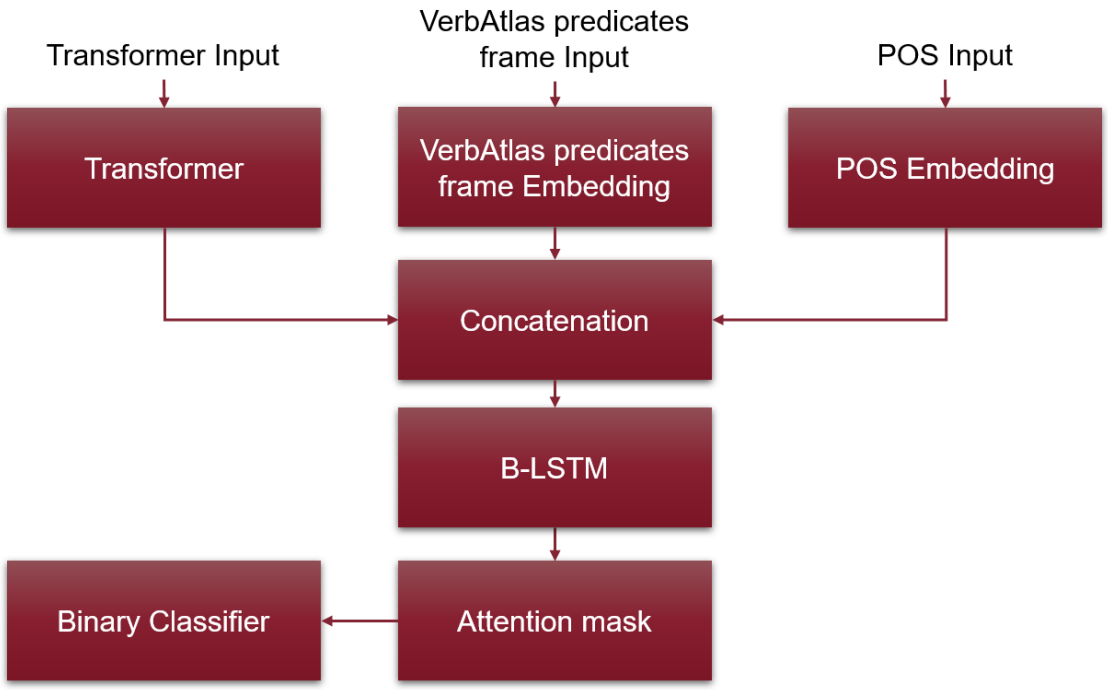
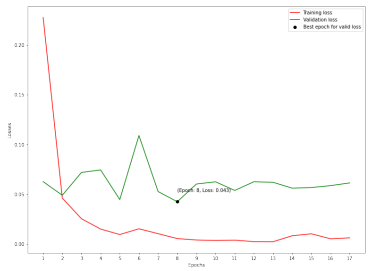
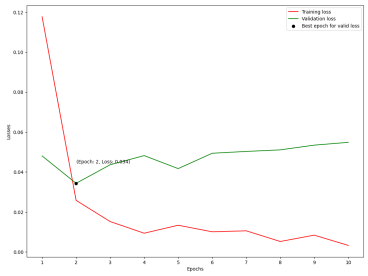


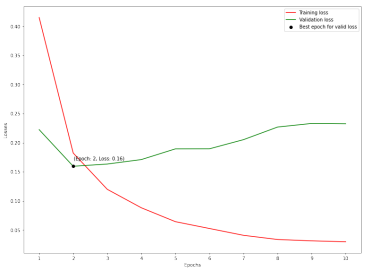
Figure 4: Event classification model architecture.



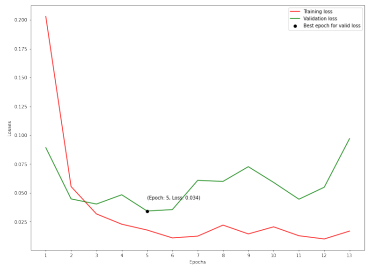
(a) Model 1 (best) losses.



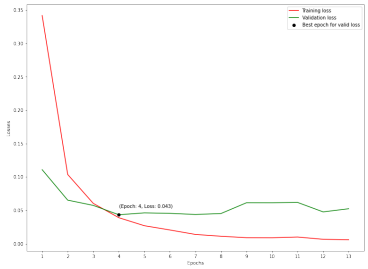
(b) Model 2 losses.



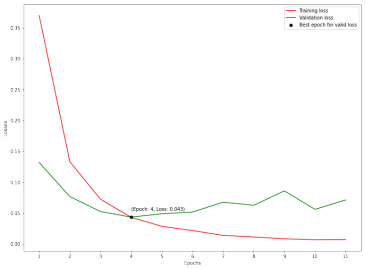
(c) Model 3 losses.



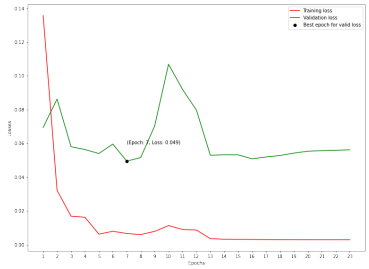
(d) Model 4 losses.



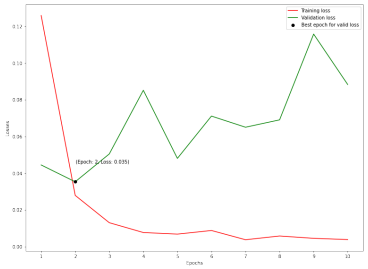
(e) Model 5 losses.



(f) Model 6 losses.

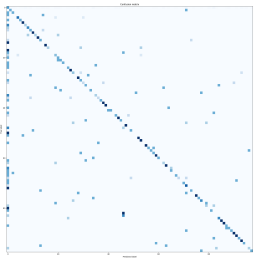


(g) Model 7 losses.

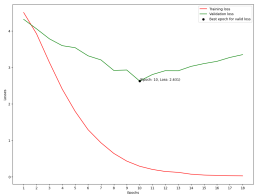


(h) Model 8 losses.

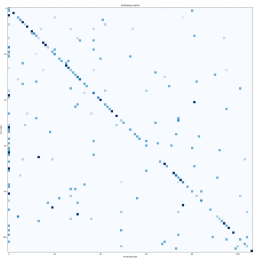
Figure 5: All identification model losses.



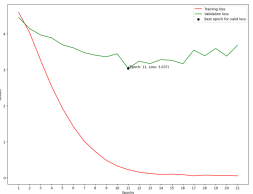
(a) Model 1 (best) confusion matrix.



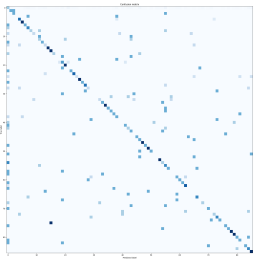
(b) Model 1 (best) losses.



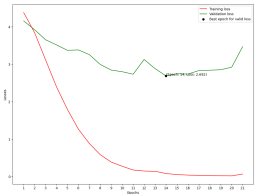
(c) Model 2 confusion matrix.



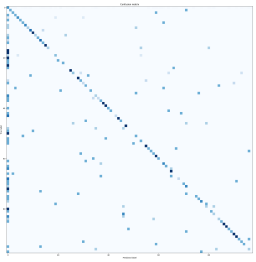
(d) Model 2 losses.



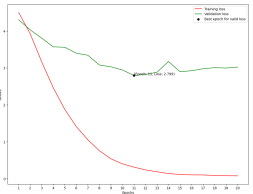
(e) Model 3 confusion matrix.



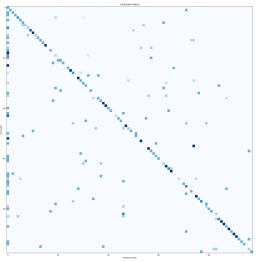
(f) Model 3 losses.



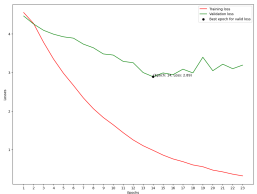
(g) Model 4 confusion matrix.



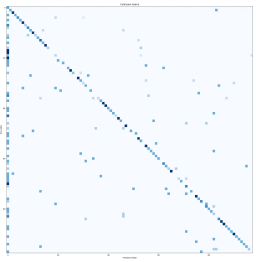
(h) Model 4 losses.



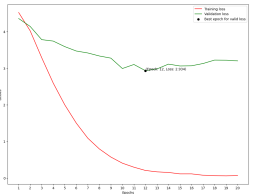
(i) Model 5 confusion matrix.



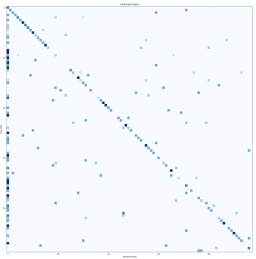
(j) Model 5 losses.



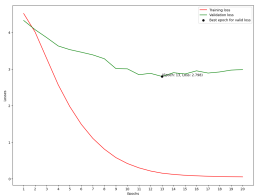
(k) Model 6 confusion matrix.



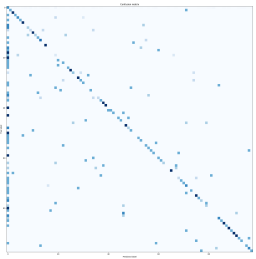
(l) Model 6 losses.



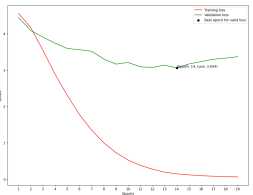
(m) Model 7 confusion matrix.



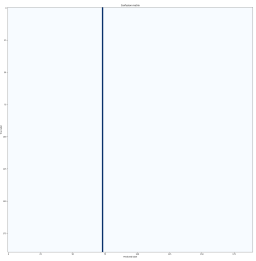
(n) Model 7 losses.



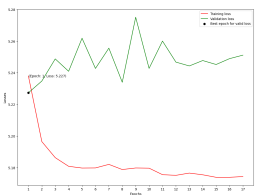
(o) Model 8 confusion matrix.



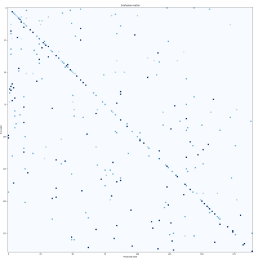
(p) Model 8 losses.



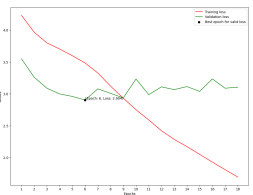
(q) Model 9 confusion matrix.



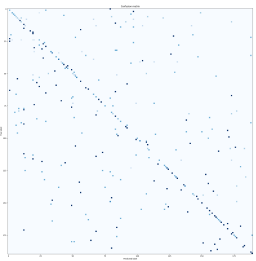
(r) Model 9 losses.



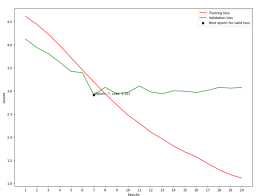
(s) Model 10 confusion matrix.



(t) Model 10 losses.



(u) Model 11 confusion matrix.



(v) Model 11 losses.

Figure 6: All classification model confusion matrices and losses.

References

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. VerbAtlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. pages 627–637.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.