# Analysis and Implementation of Multi-Inventory Word Sense Disambiguation Systems

Giada Simionato, 1822614
DIAG, Sapienza University of Rome – Via Ariosto, 25, 00185 Rome, Italy

## I. Dataset Preprocessing and Implementation Choices

In order to implement multi-inventory *Word Sense Disambiguation* (WSD) systems for three tasks, hence for *fine-grained BabelNet ids* (BN), *coarse-grained WordNet Domains* (WND) and *coarse-grained Lexicographer's ids* (LEX) predictions, in this work was primary used the *SemCor* dataset that followed the Raganato et al. format. From the `.xml` file, during a parsing phase, all information regarded lemmas, part of speech and instance ids were collected. In a first place the sentences where chosen to be composed by the words instead of their lemmas, but the size of the vocabulary, and hence of the last layer of the models, was too wide to deal with it, hence, instead of reducing the size by excluding some instances, it was chosen to use their base form. Also the information provided by the POS was considered enough important as to justify their collection, to filter the candidate synsets for each lemma in the prediction phase. After having retrieved all the data, the input and labels were shaped into tensors of fixed maximum length, by tokenizing and replacing each element, whenever required, with the corresponding indices of the vocabularies, eventually padded. Maintaining limited the memory resources usage when it came to providing the one-hot encoding version of the labels, forced to use a custom sparse categorical accuracy evaluation function during training. In order to deal with the several representations of the senses, methods for providing these mappings were implemented making easier to retrieve the format desired. Each model was trained on the whole dataset for ten epochs with a batch size of 32, an embedding size of 256, 128 hidden units per layer, the *Nesterov Adam* optimizer and a dropout rate equal to the recurrent one of 0.3. The maximum length of the input was set to 50 after a careful analysis of the distribution length of the original sentences, as shown in Fig. 1(a)

For evaluating the performances of the models implemented in this work was used the *Semeval2013* dataset, after a parsing phase aimed to retrieve the input lemmas, POS and which word was meant to be disambiguated, the data were shaped, fed into the networks and the predictions were retrieved, these data were handled by three customized prediction functions that generated the desired senses. In order to provide a quantitative measure of the performances, in this work was used the `java Scorer` inside the framework of the datasets, that produced the F1-Score.

## II. Models Description

In this section all the models analysed and implemented in this work will be explained and the performances will be discussed.

### A. Baseline, Embedding Enhancement and Skip-Connections

The model chosen as baseline for this work was a simple *Stacked Bidirectional Long-Short-Term Memory* network with a vanilla embedding layer and a *softmax* as output layer, as depicted in Fig. 2. In order to boost the performances a different embedding strategy was employed: thanks to the usage of a custom *Lambda* layer was possible to integrate, through the `Tensoflow hub` module, the state-of-the-art *ELMo* pre-trained embedding model. This change led the internal structure of the model to the one in Fig. 3. As shown in Table 1 this different strategy increased the performance of the fine-grained task of 3.5% making it the default embedding choice for the subsequent models.

Despite the baseline structure was a modified version of the first model in (Navigli et al., 2017) the first completely new feature introduced in this work was the use of *skip connections* that insert a connection between the output of the first bidirectional layer and the softmax layer for each unit, making the input of this latter the sum of the two staked Bi-LSTM layers' outputs. This additional idea, shown in Fig. 4, resulted in a slight increment of the performances, as corroborated by results in Table 2, but also in a sensible reduction of the training time. For these reasons all the future models were equipped with these connections.

### B. Architecture Diversification

In order to analyse the impact of different network architectures on the performances for all the three tasks, three different models were implemented, taking inspiration from the ones that provided best results in the WSD field, reported in (Navigli et al., 2017). The baseline endowed with ELMo embeddings and skip-connections was the first to be trained and tested for all the three tasks. The second architecture implemented in this work was the result of the augmentation of the first one with an

*attention layer* to exploit also the hidden states of the units involved in the computation of the context vector **c**. The resulting internal structure is depicted in Fig. 5. Finally the last basic architecture taken into consideration was a *sequence to sequence* model organized in an encoder-decoder structure as shown in Fig. 6. The first part is a stacked Bi-LSTM network with skip-connections, ELMo embeddings and an attention layer that outputs the vector **c**, while the second one resembles the first without the attention layer but provided of the softmax one.

All the results gathered from these models are reported in Table 3. As it's possible to notice each of these architectures can be considered the best model for a particular task: for fine-grained application the best one is the first, while for wordnet domains prediction and lexicographer's ids one are the attention and the sequence-to-sequence models respectively. This opened the possibility to deeply customize interconnected future models.

### C. Reduced-Vocabulary Models

One of the most critic feature of models in the WSD task is the handling of the output layer dimension: this has one unit per words in a joint vocabulary of all the lemmas encountered in the dataset and the ids of the senses to be determined. In coarse-grained applications, this problem is even more important due to the smaller set of categories where words can be mapped to. Whether, like in this work, the only purpose is to disambiguate specific words then how others are transformed is not relevant. For this reason is possible to reduce the vocabularies mapping all the lemmas that don't need to be disambiguated to a specific keyword in such a way that a $99.85\%$ reduction can be achieved, for example, in LEX predictions task. This led to a huge reduction of internal parameters of the networks and therefore of the training time required. Having a restricted number of classes to choose from, also slightly increased some performances, as shown in Table 4 for WND and LEX predictions.

### D. Multitask Learning with Auxiliary Losses

The links underneath the various categorizations of the senses, and their representations drove the idea to build and train models that performed different tasks simultaneously. The first of this kind was inspired by (Navigli et al., 2017) in which the fine-grained id, the part of speech and the lexicographic id were predicted at once, exploiting the knowledge of POS gathered form the dataset, that guided the training with its auxiliary loss. The schema of this model is shown in Fig. 7(a) and the internal structure of the main block is a stacked Bi-LSTM with attention layer. With this approach both the fine-grained and the LEX tasks could be addressed. A version without attention layer was also implemented and tested.

A novel approach introduced in this work is a system capable of predicting all three tasks at once working with three losses of equal weight. The schema is shown in Fig. 7(b) and the internal model is a simple stacked Bi-LSTM network. According to its structure the number of internal parameters is lower then the sum of three different networks' ones, even with the same architecture, without penalizing the number of formats that can be predicted or the performances.

The results of these three multitasking models are collected in Table 5 that highlights how the version of the first multitasking model without attention gave higher performances that its original one and how the second version became the best model for LEX predictions.

### E. Boosting Approaches

Since combining different information and tasks together in several way led to interesting results, a series of models devoted to boosting the performances for fine-grained approach using coarse-grained ones were studied and implemented in this work.

The first one, shown in Fig. 8, predicts fine-grained ids using a sequence-to-sequence network (the third in Section II-A) and, with the same input, the LEX one with an attention model (the second in Section II-A). The fine predictions were then filtered taking as candidate synsets only those belonging to the most probable lexicographic category previous gathered. The second one, in Fig. 9, exploits two networks: the first one, an attention model, takes the input and predicts LEX categories, while the second attention model takes in addition to the input the lexnames already predicted to produce the fine-grained ids. The third system, shown in Fig. 10, is made by three attention models where the first produces, from the data, the lexnames that, along with the data, become the input of the second one that produces the WordNet domains, again as input of the third one that outputs the fine-grained predictions. Another system is inspired by the first boosting approach described but the fine-grained ids are filtered taking only those coupled with synsets that belong to the WordNet domain and lexicographic category predicted by other networks starting from the same input. In this work two versions of this approach were implemented, those in Fig. 11(a) and Fig. 11(b). The first one is made by three different networks: the one for lexnames is an attention model, the one for the fine-grained ids is a sequence-to-sequence one and the one for WordNet domains is a Bi-LSTM network. The second version, instead, uses only one model for predicting all three tasks, namely the second multitasking network described in Section II-D.

All the results produced by these approaches are collected in Table 6 that shows an enhancement of the performances in each case but with picks whenever the ids were filtered during a separate step, especially when using both the coarse-grained tasks. In these cases, even if the multitask system produced a score lower than $0.9\%$ with respect to using three separate networks, it represents a

good choice due to the reduction of number of parameters involved as explained in Section II-D and therefore it was chosen as benchmark for this task.

### F. Less Data Handling

The final category of novel models analysed and implemented in this work explored the possibility of obtaining the same performances even in presence of less information. In fact, retrieving data, as the part of speech for each lemma, is one of the most cumbersome problem in NLP tasks, hence having systems capable of producing predictions without impacting on the performances, is an intriguing field. For this reason the first model, depicted in Fig. 12, is made of three networks: an attention model that converts inputs to lexnames, a Bi-LSTM one that produces POS from the same input and the last attention network that takes as input, other the data, also the lexnames and the POS predicted by the two afore-mentioned networks. The second proposed model, shown in Fig. 13, exploits a multitask approach through a Bi-LSTM network that predicts the lexnames and part of speech simultaneously and then these are fed into an attention model, along with the data to predict fine-grained ids.

The results are collected in Table 7 that shows how this latter approach gives slightly higher results and how, overall these performances are little lower than the models that used the dataset-collected data.

## III. Dataset Augmentation, Hyperparameters Tuning and Best Models

After having compared all the results obtained for each proposed model and for each task, the best architectures were highlighted to became candidates for dataset augmentation and hyperparameters tuning process. For the fine grained task the best model was the fourth described in Section II-E, while for the WordNet domains was the Bi-LSTM network and for the lexnames prediction was the second multitask approach as shown in Table 8.

To augment the SemCor dataset was used a portion of the *OMSTI* one that was respectful of the Raganato et al. format that characterized the former. Due to the high quantity of data provided by this framework, only the first 100000 sentences were taken into consideration. As in case of SemCor, it was parsed and lemmas, POS and ids were extracted and shaped to feed the models. In order to study the impact of this augmentation on the three tasks the second multitask model was chosen as benchmark and trained for $4$ epochs with the same hyperparameters, except for the sequence length that was set to $80$. The results are shown in Table 9. As it's possible to notice the fine-grained task gained a $17.4\%$ in accuracy while the coarse-grained ones decreased. For this reason it was chosen to maintain the previous mentioned models as best results for these latter tasks while was performed a deep hyperparameters tuning step for the fine-grained one.

The hyperparameters taken into consideration for the tuning phase, without the OMSTI part, were the maximum length for the input sentences, the number of hidden units, chosen among $\{128, 256, 512\}$, the dropout and recurrent dropout rates (among $\{0.2, 0.3, 0.4\}$) and the optimizer, like *Adam, Nadam and Adadelta*. For the first one, in addition to the original $50$ words limit, was chosen $80$ according to the study of the sentence length rate, whose trend for OMSTI is depicted in Fig. 1(b). The results for the grid search of seq. length, the number of hidden units and the optimizer are reported in Table 10. According to this table, the best configuration for this model was a sequence length of $80$, a hidden size of $512$ and the *Nesterov Adam* optimizer. The results for the change of the dropout rates are not shown here for the sake of brevity but they confirmed that the best choices were both $0.3$.

In summary, the three best models are:

- *Fine-grained task (BN)*: the second multitask model described in Section II-D, trained for $4$ epochs on the SemCor+OMSTI dataset, $512$ hidden units per layer, $80$ as maximum input length, $0.3$ dropout and recurrent dropout and *Nesterov Adam* as optimizer. It reached a F1-Score of $64.7\%$.
- *Coarse-grained task (WND)*: the stacked Bi-LSTM ELMo model with skip-connections, trained on SemCor for $10$ epochs, $128$ hidden units, $50$ as maximum limit and same dropout rates and optimizer as fine-grained. It reached a F1-Score of $80.4\%$.
- *Coarse-grained task (LEX)*: same model as the fine-grained task but with the hyperparameters of the WND one. It reached a F1-Score of $80.4\%$.

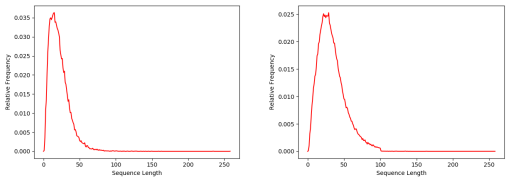
## IV. Conclusion and Future Work

In this work several models were implemented starting from changes in the embedding layer to the network architectures, exploiting the benefits of multitasking approaches, the way of combining networks in order to boost performances of one task thanks to the others and how to deal with less data available.

Despite this work covers different scenarios, it's possible to devise some future developments, like the employment of a *BERT* embedding layer, the usage of a different backoff strategy rather than MFS one or the employment of a bigger part of the OMSTI+SemCor dataset.

However the wide variety of ideas described in this work opens to a even larger pool of candidates for future enhancements in the Word Sense Disambiguation task.

## V. Tables and Figures




(a) (b)

Figure 1: Study of the distribution of length in original sentences: (a) for the SemCor dataset; (b) for the SemCor+OMSTI dataset.

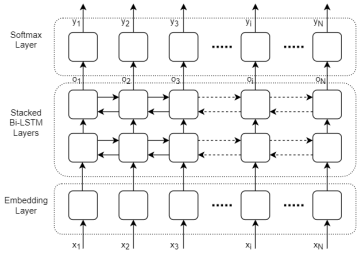



Figure 2: Baseline model: Staked Bi-LSTM network with vanilla embedding layer and a softmax as output layer. N stands for the maximum length of the input sentences.

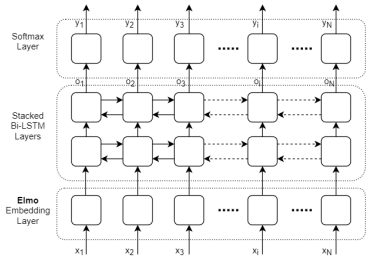



Figure 3: Model endowed with a custom ELMo embedding layer.

| | **Embedding Strategy** | |
|---|---|---|
| **Task** | Vanilla | ELMo |
| Fine-Grained (BN) | 41.8 | **45.3** |

Table 1: Embedding comparison: *vanilla* strategy represents Keras default embedding layer while *ELMo* the custom Lambda layer. Results are F1-Score on the evaluation dataset expressed as percentage.

| | **Skip-Connection** | |
|---|---|---|
| **Task** | With | Without |
| Fine-Grained (BN) | **45.6** | 45.3 |

Table 2: Skip-connections comparison: the first column shows the performance of the ELMo model with skip-connections while the second column is the same model without this feature.

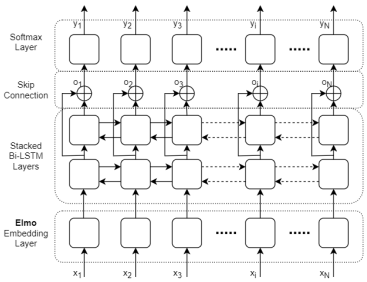



Figure 4: Baseline model enriched with the ELMo embedding layer and skip-connections to connect both layers' output to the output layer to speed up the training and slightly enhance the performances.

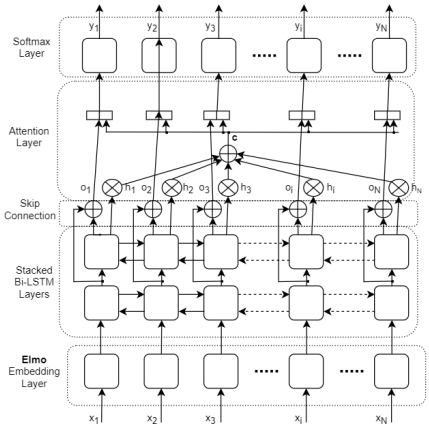



Figure 5: Second architecture: Stacked Bi-LSTM model with an attention layer.

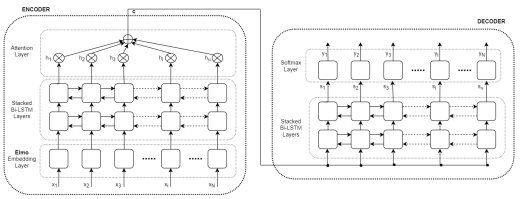


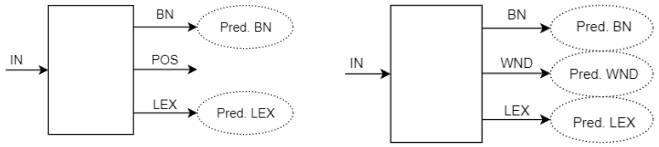

Figure 6: Third architecture: Sequence-to-sequence model structured in a attentive encoder/decoder fashion with ELMo embeddings. Skip-connections are not shown here for the sake of clarity of the schema.

| | **Task** | | |
|---|---|---|---|
| **Architecture** | F-G (BN) | C-G (WND) | C-G (LEX) |
| Bi-LSTM | 45.6 | **80.4** | 79.5 |
| Attention | 45.0 | 79.3 | **79.7** |
| Encoder/Decoder | **48.0** | 66.8 | 59.7 |

Table 3: Architecture comparison: Bi-LSTM, Attentive Bi-LSTM and Sequence-to-sequence models' performances on all three tasks. F-G (BN) stands for fine-grained, hence BabelNet ids, predictions; C-G (WND) for WordNet domains predictions; C-G (LEX) for Lexicographic's ids predictions.

| Reduced-Voc. Arch. | Task | |
|---|---|---|
| | C-G (WND) | C-G (LEX) |
| Bi-LSTM | **80.4** | **79.9** |
| Attention | 79.6 | 79.7 |
| Encoder/Decoder | 68.4 | 63.4 |

Table 4: Performances obtained by all the three architectures for the two coarse-grained tasks with a reduced vocabulary obtained by mapping all the words not-to-disambiguate to a specific keyword.




(a) (b)

Figure 7: Multitasking models structures: (a) Attentive model that predicts BN and LEX ids along with POS; (b) Stacked Bi-LSTM model that predicts BN, WND and LEX ids simultaneously.

| Mult. Model | Task | | |
|---|---|---|---|
| | F-G (BN) | C-G (WND) | C-G (LEX) |
| Mult. BN/POS/LEX | 47.2 | - | 79.3 |
| Mult. BN/POS/LEX (No Att.) | **47.3** | - | 79.6 |
| Mult. BN/WND/LEX | 45.3 | **79.7** | **80.4** |

Table 5: Multitasking model comparison: the model that provides BN-POS-LEX was implemented also in a non-attentive fashion providing the results in the second row. The novel approach outperforms the first one on two tasks out of three.

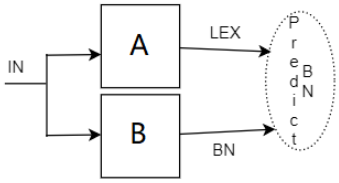



Figure 8: First boosting model: **A** is an attention model while **B** is a seq-to-seq one. Predictions are then combined in a subsequent step to filter the candidate synsets to obtain more precise results.

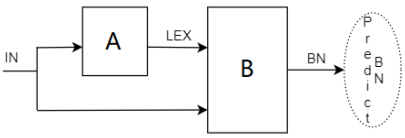



Figure 9: Second boosting model: **A** and **B** are attention models.

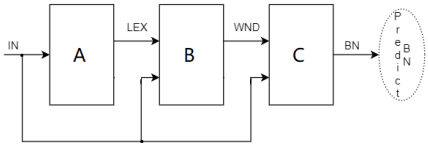
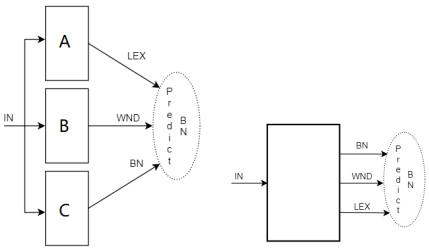



Figure 10: Third boosting model: **A**, **B** and **C** are attention models where each network is devoted to refine and channel the predictions, starting from LEX categories through WND to BN ids.




(a) (b)

Figure 11: Fourth and fifth boosting models: (a) each network predicts a task further combined in a subsequent step to filter the possible senses. **A** is an attention model, **B** is a Bi-LSTM one while **C** is a sequence-to-sequence model; (b) a multitask network of the type in Fig. 7(a) predicts all the tasks at once combined in the sequent step.

| Task | Boosting Model | | | | |
|---|---|---|---|---|---|
| | a | b | c | d | e |
| Fine-Grained (BN) | 58.6 | 48.5 | 48.3 | **63.7** | 62.8 |

Table 6: Boosting model comparison: (a) is the first boosting model (Fig. 8); (b) the second one (Fig. 9); (c) the third one (Fig. 10); (d) the fourth one (Fig. 11(a)); (e) the fifth one (Fig. 11(b)).

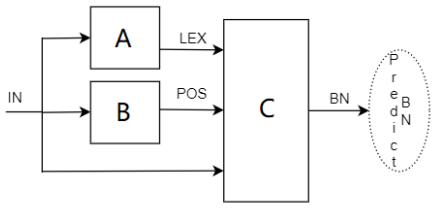



Figure 12: First model that deals with less data: **A** is an attention model, **B** is a Bi-LSTM one and **C** is another attention model.

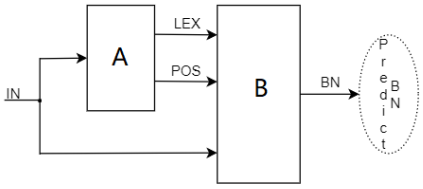



Figure 13: Second model that deals with less data: it combines the boosting idea with the multitasking technique to handle this problem. **A** is a Bi-LSTM model while **B** is an attentive one.

| | Parameters configuration | | | | | |
|---|---|---|---|---|---|---|
| **Optimizer** | a | b | c | d | e | f |
| Adam | 45.0 | 45.1 | 45.3 | 44.8 | 45.3 | 45.0 |
| Nadam | 45.3 | 45.5 | 45.9 | 45.1 | 45.8 | **46.9** |
| Adadelta | 45.4 | 45.4 | 45.2 | 45.0 | 45.6 | 45.5 |

Table 7: Hyperparameters tuning performances: all these results were obtained after a training of 5 epochs on SemCor dataset with dropout rates of 0.3 according to different configurations. (a) sequence length of 50 and hidden size of 128; (b) seq. length 50 and hidden size 256; (c) seq. length 50 and hidden size 512; (d) seq. length 80 and hidden size 128; (e) seq. length 80 and hidden size 256; (f) seq. length 80 and hidden size 512;

| | Model | |
|---|---|---|
| **Task** | a | b |
| Fine-Grained (BN) | 45.3 | **45.4** |

Table 8: Performances obtained by the models that deal with less data: (a) is model in Fig. 12; (b) is model in Fig. 13.

| | Task | | |
|---|---|---|---|
| **Model** | F-G (BN) | C-G (WND) | C-G (LEX) |
| a | 41.8 | - | - |
| b | 45.3 | - | - |
| c | 45.6 | **80.4** | 79.5 |
| d | 45.0 | 79.3 | 79.7 |
| e | 48.0 | 66.8 | 59.7 |
| f | - | **80.4** | 79.9 |
| g | - | 79.6 | 79.7 |
| h | - | 68.4 | 63.4 |
| i | 47.2 | - | 79.3 |
| j | 47.3 | - | 79.6 |
| k | 45.3 | 79.7 | **80.4** |
| l | 58.6 | - | - |
| m | 48.5 | - | - |
| n | 48.3 | - | - |
| o | **63.7** | - | - |
| p | 62.8 | - | - |
| q | 45.3 | - | - |
| r | 45.4 | - | - |

Table 9: Recap of all models' performances: (a) Baseline model; (b) Baseline + ELMo embedding layer; (c) Baseline + ELMo + Skip-Connections; (d) Baseline + ELMo + SC + Attention layer; (e) Sequence-to-sequence + ELMo + SC; (f) Bi-LSTM + ELMo + SC + restricted vocabulary; (g) Att. mod. + ELMo + SC + rest. voc.; (h) Seq.-to-seq. mod. + ELMo + SC + rest. voc.; (i) Multitask mod. (BN-POS-LEX) with Attention layer + ELMo + SC; (j) Mult. mod. (BN-POS-LEX) + ELMo + SC; (k) Mult. mod. (BN-WND-LEX) + ELMo + SC; (l) First Boosting mod. (Fig. 8); (m) Second Boost. mod. (Fig. 9); (n) Third Boost. mod. (Fig. 10); (o) Fourth Boost. mod. (Fig. 11(a)); (p) Fifth Boost. mod. (Fig. 11(b)); (q) First mod. for less data (Fig. 12); (r) Second mod. for less data (Fig. 13).

| | Task | | |
|---|---|---|---|
| **OMSTI** | F-G (BN) | C-G (WND) | C-G (LEX) |
| Without | 45.3 | **79.7** | **80.4** |
| With | **62.7** | 76.8 | 76.9 |

Table 10: Results of the model in Fig. 7(b) with only the whole SemCor dataset or with 100000 sentences of the SemCor+OMSTI augmented dataset, for 4 epochs.