Generalization AND Optimisation: A case study with SuperSequence Labelling

Akshay Aggarwal
UPV-EHU
Donostia 20013, Spain
akshayantimatter@gmail.com

Abstract

In this paper, we try to see the effects of supersequence labelling being considered as a multi-class NER problem. We understand that the best-performing models in NER task are still not generalizing well enough to be useful for transfer learning, for example. We combine two freely available datasets to create a new dataset where the test data is more difficult than train+dev data, forcing the models to learn general features, rather than the localised ones. We also try to understand if the different embeddings, having been trained in a different manner, offer some advantages/disadvantages in the task of supersequence WSD. The experiments were not completed, and so the final results of the evaluation are not clear.

1 Introduction

SuperSequence Labelling, or Supersense Sequence Labelling is not a new task in any way. Linked to semantics, there are arguments debating the technology can help us come up with better parsers, Named-Entity recognisers, WSD, etc. In essence, if we capture the semantic properties of the underlying words, and then process the sentence like a sequence, we should be able to push up the results for different language technologies.

Working with semantics is not an easy task, nonetheless. For English, there exists semantically annotated data. Some of the examples include WordNet, FrameNet, PropBank, etc. However, for most languages out there, there are no such existing resources. There is an added question of how do we know that the system is understanding the semantics, and is not just playing with for example, properties, vectors, etc. to return to us the right answer(s). There have been tasks that check the semantic inference of a system. For example- Question Answering System, Text Summarisations, WSD, etc. The task with WSD is one of the oldest ones in literature, and can be defined as the task of tagging data with the WordNet labels (categories). This is usually considered as a tagging, and a classification problem.

Also referred to as extension of NER systems, the problem has been approached in multiple ways. However, most of the results verify their findings on either of SemCor, SemEval data, among others. For NER tasks, the comparison results are on test set of CONLL-2003 shared task, for German, and English languages. This poses a problem. We are very likely to saturate the dataset, since we are achieving better results on the data. A question that needs to be askeds if we are improving the models or just setting up a higher SOTA? Both the tasks are different, and often show little to a medium correlation.

In this paper, we try to ask again the two questions on the models which have been shown in past to be SOTA at certain points. We compare SOTAs from different points in time, and then we devise a new dataset that forces the model(s) to learn general features. If the model is learning local features, and can't generalize well, it would essentially result in a lower F1 score in the setting. The dataset used for this task is detailed upon in Section 2.1. We also try and seek to understand if the different word-embeddings, having been trained in different approaches, offer some added nuances to the task of

general learning.

The rest of this document is organized as follows. Section 2 elaborates on the experimental setup and the experiments conducted, with the results of the experiments being presented in Section 3. We discuss on some of the results and discuss on the future scope of the paper in Section 4, after which we finally conclude in Section 5.

2 Experiments

2.1 Data

The data has been split into train, dev and test sets. Of these, we select test set from another corpus, so as to make sure the model generalizes and there is a lower chance of the data meeting the statistical model for training and dev data.

The training (train + dev) dataset is formed by Semcor corpus obtainable from Kaggle[1]. Semcor is divided as follows:

- SEM: brown1 and brown2 corpus; contains annotations for nouns, verbs, adjectives, and adverbs.
- SEMv: brownv corpus; contains annotations for just verbs.

The test data is the freely available Seneval-3 English Allwords (SE3) data. This data contains of 2 WSJ articles, and a fiction excerpt from Brown corpus.

For NER data, there exist two popular formats of annotation. The IOB (In, Out, Begin) tagset marks the named-entities in the format so that there is a clear division of where the named-entity chunk begins (tag B), which units form a part of the chunk (tag I), and which units don't form part of any chunk whatsoever.

Introduced as an improvement to IOB tagset, IOBES (In, Out, Begin, End, Single) tagset offers more fine-tuned annotation of the data. The singleton chunks of named-entitites are marked explicitly (tag S), and there is a clear marker for where the current multi-word chunk ends (tag E).

The entirety of the data (test + dev + train) is converted first into IOB format, and then also IOBES format. Since it's unclear which format offers more advantage to the system, we define an experiment (Section 2.2.1) to resolve which tagset we would like to use. For each of the IOB or IOBES tagged versions of the data, we perform a few pre-processing steps as follows-

- Wordnet senses are substituted by the corresponding supersense.
- The supersense "noun.Tops" was substituted by tag O.

Parameter	SEM	SEMv	SE3
Sentences	20,138	17,038	300
Tokens	434,774	385,546	5,630
Supersenses (SS)	135,135	40,911	1,617
Verbs	47,710	40,911	725
Nouns	87,425	00	892
Avg-polysemy N-WS	4.41	4.33	4.66
Avg-polysemy N-SS	2.75	2.66	2.86
Avg-polysemy V-WS	10.87	11.05	11.17
Avg-polysemy V-SS	4.11	4.16	4.20

Table 1: Datasets and the statistical counts

Note: Avg = Average; WS = Word Sense; N = Noun; V= Verb

The datasets, and the different counts of each dataset are reflected in Table 1. The dataset is freely available to be downloaded from different sources. However, the last column shows us that the test data is definitely more polysemous than training data. This implies that the task would be a difficult one, as the model needs to learn features as general as possible. We want to check given a trained model, how well the generalization can be achieved, even with limited data.

2.2 Models that generalize well

We started our experiments with the perceptron-based model as defined in (Ciaramita and Altun, 2010) on the given data. The model in question contains hand-annotated rules, and combining them with a hidden Markov Model (HMM). We shall hereafter define this model as the baseline model.

Next, we train the models defined in (Lample et al., 2016), (Huang et al., 2015) and (Ma and Hovy, 2016) on the given data, without parameter optimization of the hyper-parameters. The hyper-parameters and the model definitions were implemented using an existing github repository[2]. It is

[1] https://www.kaggle.com/nltkdata/semcor-corpus

[2] https://github.com/guillaumegenthial/tf_ner

Model Name	IOB	IOBES
(Huang et al., 2015)	61.92	62.31
(Lample et al., 2016)	60.30	61.80
(Ma and Hovy, 2016)	**63.03**	**62.87**

Table 2: Models and their F1 scores, when run **without** hyper-parameter tuning on different annotation scheme

Model Name	IOB	IOBES
Baseline	**69.46**	**69.46**
(Huang et al., 2015)	66.31	67.68
(Lample et al., 2016)	67.38	67.48
(Ma and Hovy, 2016)	66.51	66.73

Table 3: Models and their F1 scores, when run **with** hyper-parameter tuning on different annotation scheme

important to note that the scores in the table differ significantly from the official scores reported in the corresponding publications. However, we are interested in models that generalize well, rather than building and working on a genre-sensitive language model. And thus, with the given dataset, it's clear why the models' performance is lower than in genre-sensitive framework as reported in the publications.

2.2.1 IOB vs IOBES

While both set of data (IOB and IOBES) have their own advantages, IOBES offers more fine-tuned annotation at the cost of more tags to be generated. It's easier to argue in theory which works better, the data was tagged in both formats, and the results evaluated with and without hyper-parameter tuning, as can be seen in Table 2 and Table 3 respectively.

The values of tuned hyper-parameters (grid-search based tuning) for the different annotation schemes are illustrated in Table 4. We decide to continue the rest of the experiments with the IOBES tagset, since it allows for more precision of the tags, annotating it better.

Model Name	IOB	IOBES
(Huang et al., 2015)	(0.20, 150)	(0.45, 300)
(Lample et al., 2016)	(0.25, 175)	(0.30, 175)
(Ma and Hovy, 2016)	(0.30, 125)	(0.25, 125)

Table 4: Tuned hyper-parameters values.

Note: (a,b) refers to an ordered pair of dropout-rate, and the size of word-LSTM, respectively.

Parameter	Value
RNN Layers	1
Hidden Units	400
Dropout	0.50
CRF	Yes
Learning Rate	0.20
Maximum Epochs	15
Early Stopping	No
Mini-Batch Size	32

Table 5: Base Model Description

2.3 Effect of Embeddings

As elaborated before, we decided to continue our experiments with the IOBES tagset after noticing the results from Table 3. Also, IOBES is supposed to provide better and more fine-tuned annotation. And that is definitely one of the factors we take into consideration when we continue with the experiments.

To study how we can generalize a model, we build a RNN-based model. For the purpose of the study, we build a single-layer RNN model, with hidden units size of 400. We add a dropout rate of 0.50, and an epoch-wise learning rate decay, with the initial learning rate set at 0.20. The model features can be seen in Table 5.

With the model in place, we start with the selection of different embeddings. The models used in the previous section used GloVe embeddings (2014), set with a dimensionality of 300. We do not fix the dimensionality of the used embeddings, rather allowing the model to select the parameter itself. This is in contrast to the approach taken by (Iacobacci et al., 2016) where they also focus on fixing dimension of embeddings, to analyse them better. We argue that since the dimension of the word-embeddings is a hyper-parameter that depends on task. As such, it's fair for the model to utilise as many dimensions as it requires, rather than fixing it as a parameter.

For the base model that we defined earlier, we focus on a few distinct approaches.

2.3.1 Word-Embeddings only

In this series of experiments, we don't use character, or token-based embeddings. We focus ourselves on the word-embeddings only. We take a look at the embeddings as listed in Table 6. For each embedding listed in the table, we use it as an input to the base model, and report on the F1 scores.

Word Embedding(s)	Reference
GloVe	(Pennington et al., 2014)
ElMo	(Peters et al., 2018)
BERT	(Devlin et al., 2018)
Flair	(Akbik et al., 2018)
ElMo + BERT	Stacked Embeddings
ElMo + Flair	Stacked Embeddings
Flair + BERT	Stacked Embeddings
ElMo + Flair + BERT	Stacked Embeddings

Table 6: Word-Embeddings used for base model

2.3.2 Character-Embeddings

To add to the effect of Word-Embeddings in previous subsection, we extend the models as from before by adding a character-level embedding over the word-embeddings as mentioned in Table 6. The character embeddings are not pre-trained but are generated by the model itself. Again, we look at the F1 scores for how well the models perform.

2.3.3 Comparison against SOTA

The state-of-the-art (SOTA) model as defined by (Akbik et al., 2019) on NER task can be found online [3]. We train the model using the same parameters, on the training and dev set, and evaluate it on our data. We call this experimental setup as ner_sota henceforth.

3 Results

Owing to a large number of models that required training over large amount of time, the results are not yet available. The models are being trained on nVidia 1080Ti GPU, however the individual epochs also take time to evaluate, hence lack of results.

4 Discussion and Future Scope

4.1 Discussion

In this paper, we tried to analyse how the different models that offer 90%+ F1 scores on NER task, still fail to generalize as can be seen with the data in Table 3. There can be further comments on how the embeddings affect the results. Intuitively, humans do not process the sentence as an entire chunk and so BERT embeddings should not be the best fit for this case. However, given the success of the embeddings in different NLP tasks, this is a claim that will be interesting to investigate into.

Another intuition says that RNNs process the data sequentially, and so should be a better fit for super-sequence labelling. However, given how CNNs can perform better for a given task, this intuition needs to be verified. Again, it could be the case that RNNs process the data better, but somehow CNNs perform the classification task better.

4.2 Future Scope of the Paper

In future, once we are done with the training of models, and can see the effect of embeddings; we can take a look at understanding why the best embedding is the best in the task. We also seek to play around with additional hyper-parameters to be able to bypass the best-performing baseline model with hand crafted features.

5 Conclusion

We understand that the best-performing models in NER task are still not generalizing well enough. With the scores reported on CONLL-2003 shared task data, we are at a danger of saturating the dataset. It's also not optimal to train large models that can't be generalised well, since they would need to be trained for individual datasets.

We try to understand the behaviour of different word-embeddings to understand how they can be used to generalize the problem better. We also hope this new task would help other researchers to train their models in a more general way, and not overfitting on the given genre.

[3] https://github.com/zalandoresearch/flair/blob/master/resources/docs/EXPERIMENTS.md#conll-03-named-entity-recognition-english

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 724728.

Massimiliano Ciaramita and Yasemin Altun. 2010. Broad-coverage sense disambiguation and information

extraction with a supersense sequence tagger. page 594.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (Mlm).

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. (2003):897–907.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. pages 260–270.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.