Concept Tagging for Natural Language Understanding: Two Decadelong Algorithm Development

Jacopo Gobbi
University of Trento
Trento, Italy
jacopo.gobbi
@studenti.unitn.it

Evgeny A. Stepanov
VUI, Inc.
Trento, Italy
eas@vui.com

Giuseppe Riccardi
University of Trento
Trento, Italy
giuseppe.riccardi
@unitn.it

Abstract

English. Concept tagging is a type of structured learning needed for natural language understanding (NLU) systems. In this task, meaning labels from a domain ontology are assigned to word sequences. In this paper, we review the algorithms developed over the last twenty five years. We perform a comparative evaluation of generative, discriminative and deep learning methods on two public datasets. We report on the statistical variability performance measurements. The third contribution is the release of a repository of the algorithms, datasets and recipes for NLU evaluation.

Italiano. *L'annotazione automatica dei concetti è un tipo di apprendimento strutturato necessario per i sistemi di comprensione del linguaggio naturale (NLU). In questo processo le etichette di un'ontologia di dominio sono assegnate a sequenze di parole. In questo articolo esaminiamo gli algoritmi sviluppati negli ultimi venticinque anni. Eseguiamo una valutazione comparativa dei metodi di apprendimento generativo, discriminatorio e approfondito su due set di dati pubblici. Il secondo contributo é un'analisi della variabilitá delle misure di valutazione. Il terzo contributo è il rilascio di un archivio degli algoritmi, dei sets di dati e delle ricette per la valutazione dell'NLU.*

1 Introduction

The NLU component of a conversational system requires an automatic extraction of concept tags, dialogue acts, domain labels and entities. In this paper we describe and review the algorithm development of the concept tagging (a.k.a. slot filling or entity extraction) task. It aims at computing a sequence of concept units, $C = c_1..c_M$, from a sequence of words in natural language, $W = w_1..w_N$. The task can be seen as a structured learning problem where words are the input and concepts are the output labels. In other words, the objective is to map a sentence (utterance) "*I want to go from Boston to Atlanta on Monday*" to the sequence of domain labels "`null null null null null fromloc.city null toloc.city null depart_date.day_name`", that would allow to identify, for instance that *Boston* is a *departure city* . Difficulties may arise from different factors, such as the variable token span of concepts, the long-distance word dependencies, a large and ever changing vocabulary, or subtle semantic implications that might be hard to capture at a surface level or without some prior context knowledge.

Since the early nineties (Pieraccini and Levin, 1992), the task has been designed as a core component of the natural language understanding process in domain-limited conversational systems. Over the years, algorithms have been developed for generative, discriminative and, more recently, for deep learning frameworks. In this paper, we provide a comprehensive review of the algorithms, their parameters and their respective state-of-the-art performances. We discuss the relative advantages and differences amongst algorithms in terms of performances and statistical variability and the optimal parameter settings. Last but not least, we have designed and provided a repository of the data, algorithms, implementations and parameter settings on two public datasets. The GitHub repository[1] is intended as a reference both for practitioners and for algorithm development researchers.

With the conversational AI gaining popularity, the area of NLU is too vast to mention all relevant

[1] www.github.com/fruttasecca/concept-tagging-with-neural-networks

or even recent studies. Moreover the objective of this paper is to benchmark an important subtask of NLU, concept tagging used by advanced conversational systems. We benchmark generative, discriminative and deep learning approaches to NLU, the work is in-line with the works of (Raymond and Riccardi, 2007; Mesnil et al., 2015; Bechet and Raymond, 2018). Unlike previously mentioned comparative performance analysis, in this paper, we benchmark deep learning architectures and compare them to a generative and traditional discriminative algorithms. To the best of our knowledge, this is the first comprehensive comparison of concept tagging algorithms at this scale on public datasets and shared algorithm implementations (and their parameter settings).

2 Algorithms

Among the algorithms considered for benchmarking, we include a representative from the generative class, the weighted finite state transducers (WFSTs), and two discriminative algorithms: Support Vector Machines (SVMs), Conditional Random Fields (CRFs), and a set of base neural networks architectures and their combinations.

Weighted Finite State Transducers[2] cast concept tagging as a translation problem from words to concepts (Raymond and Riccardi, 2007), and usually consist of two components. The first component transduces words to concepts based on a score that can be either induced from data or manually designed; the second component is a stochastic conceptual language model, which re-scores concept sequences. The two components are composed to perform sequence-to-sequence translation and infer the best sequence using Viterbi algorithm.

Support Vector Machines (SVM) are used within Yamcha tool (Kudo and Matsumoto, 2001) that performs sequence labeling using forward and backward moving classifiers. Automatic labels assigned to preceding tokens are used as dynamic features for the current token's label decision.

Conditional Random Fields (CRF)[3] (Lafferty et al., 2001) is a discriminative model based on a dependency graph G and a set of features. Each feature f_k has an associated weight λ_k. Features are generally hand-crafted and their weights are learned from the training data. Additionally, we experiment with word embeddings as additional features for CRFs (CRF+EMB).

Recurrent Neural Networks (RNN). The first neural network architecture[4] we have considered is an Elman RNN (Elman, 1990; Übeyli and Übeyli, 2012). In RNN, a hidden state depends on the current input and the previous hidden state. The output (label), on the other hand, depends on the new hidden state.

Long-Short Term Memory (LSTM) RNNs (Hochreiter and Schmidhuber, 1997) try to tackle the vanishing gradient problem by introducing a more complex mechanisms to address information propagation and deletion, with the cost of a more complex model with more parameters to train due to the system of gates it uses. The memory of the model is represented by the cell state and the hidden state, which also represents the output for the current token. We experimented with a simple LSTM, an LSTM which receives as input the word embedding concatenated with character embeddings obtained through a convolutional layer (Józefowicz et al., 2016) (LSTM-CHAR-REP), and an LSTM with pre-trained embeddings and dynamic embeddings learned from training data (LSTM-2CH). In LSTM-2CH two separate LSTM modules run in parallel and their outputs are concatenated for each word. Similar to the rest of the deep learning models, the output is then fed to a fully connected layer to map every token to the concept tag space.

Gated Recurrent Units (GRU) (Cho et al., 2014) use a reset and an update gate, which are two vectors of weights that decide what information is deleted (or re-scaled) from the current hidden state and how it will contribute to the new hidden state, which is also the output for the current input. Compared to the LSTM model, this allows to train fewer parameters, but introduces a constraint on memory, since it is also used as an output.

Convolutional Neural Networks (CONV) (Majumder et al., 2017; Kim, 2014) consider each sentence as a matrix of shape (# words in sentence, size of embedding) for convolution using kernels of different sizes to pass over the input sequence token-by-token, bigram by bigram and trigram by trigram. The result of convolution is used as a

[2] We use OpenFST (http://www.openfst.org) and OpenGRM (http://www.opengrm.org) libraries.

[3] We use CRFSUITE (Okazaki, 2007) implementation of CRFs in out experiments.

[4] All neural architectures are implemented within the PyTorch framework (https://pytorch.org)

starting hidden memory for a GRU RNN. GRU RNN is used on embedded tokens and starts with the information on the sequence at a global level.

FC-INIT is similar to CONV. The difference is in the pre-elaboration of the hidden state, which is done by fully connected layers elaborating on the whole sequence.

ENCODER architecture (Cho et al., 2014) casts the problem as a sequence-to-sequence translation and consists of two GRU RNNs. Encoder, the first GRU RNN, encodes the input sequence to a fixed vector (the hidden state). Decoder, another GRU RNN, uses the output of the encoder as a starting hidden state. At each step, the decoder receives the label predicted at the previous step as an input, starting with a special token.

ATTENTION architecture is similar to ENCODER with the addition of an attention mechanism (Bahdanau et al., 2014) on the outputs of the encoder. This allows the network to focus on a specific parts of the input sequence. The attention weights are computed with a single fully connected layer that receives as input the embedding of the current word concatenated to the last hidden state.

LSTM-CRF (Yao et al., 2014; Zheng et al., 2015) is an architecture where the LSTM provides class scores for each token, and the Viterbi algorithm decides on the labels of the sequence at a global level using bigrams and transition probabilities that are trained with the rest of the parameters. We also experimented with a variant that considers character level information (LSTM-CRF-CHAR-REP).

3 Corpora

The evaluation of algorithms is performed on two datasets. The Air Travel Information System (**ATIS**) dataset consists of sentences from users querying for information about flights, departure dates, arrivals, etc. The training set consists of 4,978 sentences, while there are 893 sentences that constitute the test set. The average length of a sentence is around 11 tokens, and there are a total of 127 unique tags (with IOB prefixes). Moreover, the large majority of tokens missing an embedding are either numbers or airport/basis/aircraft codes. The training set has a total of 18 types missing an embedding, and the test set has 9.

Model	**Parameters**	**# Params**	F_1
WFST	order 4, kneser ney	(7907 states, 842178 arcs)	82.96
	order 4, kneser ney	(4124 states, 76000 arcs)	93.08
SVM	(4, 4) window of tokens, (-1, 0) of POS tag and prefix. Postfix and lemma of current word. Previous two labels.	10364	83.74
	(6, 4) window of tokens, (-1, 0) of prefix and postfix. Previous two labels .	16361	92.91
CRF	(4, 4) window of token, (-1, 0) of POS tag and prefix. Postfix and lemma of current word. Previous + current word conjunction, current + next word conjunction. Bigram model.	1,200K	83.80
	(6, 4) window of tokens, (-1, 0) of prefix. Postfix of current word. Previous + current word conjunction. Bigram model.	2,201K	93.98
CRF+EMB	all above + (4, 4) word embs + current token char embeddings	1,390K	85.85
	all above + (6, 4) word embs + current token char embeddings	3,185K	94.00

Table 1: F_1-scores for the WFST, SVM and CRF (with and without embeddings) algorithms on the MOVIES (top row) and ATIS (bottom row) datasets.

The second corpus (**MOVIES**)[5] was produced from NL2SparQL (Chen et al., 2014) corpus semi-automatically aligning SPARQL query values to utterance tokens. The dataset follows the split of the original corpus having 3,338 sentences (with 1,728 unique tokens) and 1,084 sentences (with 1,039 tokens) in the training and test sets, respectively. The average length of a sentence is 6.50 and the OOV rate is 0.24. There are 43 concept tags in the dataset. Given the Google embeddings, once we consider every number as a class *number*, we obtain 66 token types without an embedding for the training set and 26 for the test set.

4 Performance Analysis

One of our first observations is the fact that models such as WFST, SVM and CRF yield competitive results with simple setups and few hyperparameters to be tuned. The training of our deep learning models and the search of their hyperparameters would have been unfeasible without dedicated hardware, while it took a fraction of the effort for WFST, SVM and CRF. Moreover, adding word embeddings as features to the CRF allowed it to outperform most of the deep neural networks.

[5] `https://github.com/esrel/NL2SparQL4NLU`

Model	hidden	epochs	batch size	lr	drop rate	emb norm	# of params	min F_1	avg F_1	best F_1
RNN	200	15	50	0.001	0.30	4	1,264K	81.00	82.55	83.96
	400	10	50	0.001	0.25	2	580K	91.80	93.79	95.03
LSTM	200	15	20	0.001	0.70	6	1,505K	82.67	83.76	84.57
	200	15	10	0.001	0.50	8	675K	87.82	94.53	95.36
LSTM-CHAR-REP	400	20	20	0.001	0.70	4	2,085K	82.00	84.28	85.41
	400	15	10	0.001	0.50	6	1,272K	81.00	94.19	95.39
LSTM-2CH	200	20	15	0.001	0.30	8	1,310K	81.22	82.68	83.76
	400	10	100	0.010	0.70	6	1,022K	93.10	94.61	95.38
GRU	200	20	20	0.001	0.50	4	1,424K	76.56	84.29	85.47
	100	15	10	0.005	0.50	10	446K	91.53	94.28	95.28
CONV	200	20	20	0.001	0.50	4	2,646K	84.05	85.02	86.17
	100	15	10	0.005	0.00	2	625K	91.51	94.22	95.38
FC-INIT	100	30	20	0.001	0.30	4	2,805K	82.22	83.93	84.95
	400	15	50	0.010	0.25	4	7,144K	87.39	94.67	95.39
ENCODER	200	30	20	0.001	0.70	4	1,559K	71.25	76.39	79.00
	200	25	5	0.001	0.70	6	730K	70.01	78.16	80.85
ATTENTION	200	15	20	0.001	0.30	4	1,712K	71.86	79.77	82.67
	200	25	5	0.001	0.25	10	894K	92.47	94.09	94.98
LSTM-CRF	200	10	1	0.001	0.70	6	1,507K	84.75	**86.11**	87.47
	400	15	10	0.001	0.50	6	1,200K	94.39	94.72	95.01
LSTM-CRF-CHAR-REP	200	15	1	0.001	0.70	8	1,555K	85.07	86.08	87.05
	200	20	5	0.001	0.50	4	740K	94.45	**94.91**	95.12

Table 2: All models are bidirectional and have been trained with unfrozen Google embeddings, except for CONV and LSTM-2CH. Min, average and best F_1 scores are obtained training the same model with the same hyperparameters, but different parameter initializations. Averages are from 50 runs for MOVIES and 25 for ATIS. For each architecture, the first row reports F_1-score for the MOVIES dataset and the second for ATIS. Hyperparameter search has been done randomly over ranges of values taken from published work. The number of parameters refers to the network parameters plus the embeddings, when those are unfrozen. Given a hidden layer size X reported in **hidden** column, each component in the bidirectional architecture would have a hidden layer size of $X/2$. Similarly, each of the two LSTM components in the LSTM-2CH model would have $X/2$ as a hidden layer size; and each bidirectional component would thus have a hidden layer size equal to $X/4$.

We attribute this to two factors: (1) since these models, unlike neural networks, do not learn feature representation from data, they are simpler and faster to train; and, most importantly, (2) these models usually perform global optimization over the label sequence, while neural networks usually do not. Augmenting neural networks with CRF is not expensive in terms of parameters. Having a CRF component on top of an LSTM increments the number of parameters up to the square of the tag-set size (about 2,500 for the MOVIES dataset), and provides the best performing model.

There seems to be no strong correlation between the number of parameters and the variance of a model performance with respect to the random initialization of its parameters. This is surprising, given the intuition that more parameters can potentially lead to a lower probability of being stuck in a local minima. The case may be that different initializations lead to different training times required to get to good local minimas.

4.1 Statistical Significance Testing

The best performing algorithms in our experimental settings are LSTM-CRF and LSTM-CRF-CHAR-REP; however, they are not very far from CRF+EMB and CRF algorithms. In order to compare the performances in terms of statistical significance, we perform Welch's unequal variances t-test (Welch, 1947), which, compared to more popular Student's t-test, does not assume equal variances. The choice of test is motivated by the observation that neural architectures generally yield higher variances than, for instance, CRF.

The performances are compared on 10-fold cross-validation outputs on the training set for both ATIS and MOVIES datasets. Due to the higher variance of neural network architectures, a better way to test would be to perform many runs with different random initializations for each fold, and take the average of these results; however, such a procedure is computationally very demanding.

ALGORITHMS	*CRF*	*CRF-EMB*	*LSTM-CRF*	*LSTM-CRF-CHAR-REP*
MOVIES				
CRF	■			
CRF-EMB	*	■		
LSTM-CRF	*		■	
LSTM-CRF-CHAR-REP	*			■
ATIS				
CRF	■			
CRF-EMB		■		
LSTM-CRF	*		■	
LSTM-CRF-CHAR-REP	*	*		■

Table 3: Results of statistical significance testing using Welch's t-test for MOVIES and ATIS datasets. Algorithms on rows with statistically significant differences in performance with $p < 0.05$ in comparison to the algorithms on columns are marked with '*'.

The results of the statistical significance testing are reported in Table 3. For the MOVIES dataset, all the compared models (CRF-EMB, LSTM-CRF, LSTM-CRF-CHAR-REP) significantly outperform the CRF model with $p < 0.05$. However, these models do not yield statistically significant differences among themselves. Specifically, using embeddings with CRF (i.e. CRF-EMB) produces statistically significant differences in performance on top of CRF. Using CRF with LSTM, even though produces better average F_1 than CRF-EMB, the gain is not statistically significant, irrespective of the type of embeddings used.

For the ATIS dataset, on the other hand, use of embeddings with CRF does not yield statistically significant differences with respect to plain CRF. Neural architectures (LSTM-CRF and LSTM-CRF-CHAR-REP), on the other hand, do produce statistically significant difference in performance in comparison to CRF. Moreover, unlike for MOVIES dataset, the use of character embeddings in LSTM-CRF architecture significantly outperforms the CRF-EMB model.

4.2 Error Analysis

Both MOVIES and ATIS datasets have imbalanced distribution of concept labels. The imbalanced distribution of labels is known to affect the performance of the minority classes. Consequently, we correlate the distribution of labels in the training set to the percent of their mis-labeling in the test set (by any model). As expected, the mis-labeling chance is inversely correlated to the percentage of instances the label has in the training set (e.g. given that a label amounts to less than 1% of a dataset, it usually has a mis-labeling chance greater than 10%). For both datasets, the Kendall rank correlation coefficients (Kendall, 1938) are approximately 0.6.

Independent of the distribution, there are certain concepts that are mis-labeled more often. For example, this is the case for **producer name**, **person name**, and **director name** in MOVIES, and **city name**, **state name**, and **airport name** in ATIS. It is not surprising given that these concepts share the values (e.g. the same person may be an actor, director, and producer) and frequently lexical contexts.

Supporting the observations in (Bechet and Raymond, 2018) for ATIS, some errors stem from inconsistent labeling. For instance, in the MOVIES dataset, "*classic cars*" is mapped to "`O O`", but "*are there any documentaries on classic cars*" appears as "`O O O B-movie.genre O B-movie.subject I-movie.subject`".

5 Conclusion

One of the main outcomes of our experiments is that sequence-level optimization is key to achieve the best performance. Moreover, augmenting any neural architecture with a CRF layer on top has a very low cost in terms of parameters and a very good return in terms of performance. Our best performing models (in terms of average F_1) are LSTM-CRF and LSTM-CRF-CHAR-REP. In general we may say that adding a sequence level control to different type of NN architectures leads to very good model performances. Another important observation is the variance of performance of NN models with respect to initialization parameters. Consequently, we strongly believe that this variability should be taken into consideration and reported (with the lowest and highest performances) to improve the reliability and replicability of the published results.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Frederic Bechet and Christian Raymond. 2018. Is ATIS too shallow to go deeper for benchmarking spoken language understanding models? In *Interspeech*.

Yun-Nung Chen, Dilek Hakkani-Tür, and Gokan Tur. 2014. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 242–247. IEEE.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Jeffrey L. Elman. 1990. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79, March.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs). *URL http://www. chokkan. org/software/crfsuite*.

Roberto Pieraccini and Esther Levin. 1992. Stochastic representation of semantic structure for speech understanding. *Speech Communication*, 11(2):283 – 288. Eurospeech '91.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*, pages 1605–1608. ISCA.

Elif Derya Übeyli and Mustafa Übeyli. 2012. Case studies for applications of elman recurrent neural networks.

B. L. Welch. 1947. The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34(1-2):28–35.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong(Shiao-Long) Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *ICASSP 2014*. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), January.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. 2015. Conditional random fields as recurrent neural networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1529–1537, Washington, DC, USA. IEEE Computer Society.