MINISTRY OF EDUCATION, YOUTH AND SPORTS

norway grants

The Research Council of Norway

D5.1, 5.3 & 5.4: Semantic search interface v.1–3

Authors:

Björn Gambäck
NTNU
gamback@idi.ntnu.no

Utpal Kumar Sikdar
NTNU
utpals@idi.ntnu.no

Lars Bungum
NTNU
larsbun@idi.ntnu.no

Grant Agreement Number	7F14047
Project Acronym	HaBiT
Project Title	Harvesting big text data for under-resourced languages
Deliverable Title	**D5.1, 5.3 & 5.4: Semantic search interface v.1–3**
Responsible Partner	**Björn Gambäck, NTNU**
Dissemination Level	Public
Due Delivery Date	30 April 2017 (+30 days)
Actual Delivery Date	1 June 2017
Status	Final, v1.0

Principal Investigator	**Karel Pala**
Project Promoter	**Masaryk University**
Tel	+420 549 49 5616
E-mail	pala@fi.muni.cz
Project Website Address	www.habit-project.eu

Table of Contents

1 Introduction

The main background for the present deliverable is given in the deliverable "D1.1.4 & D5.2: Semantic content matching — Module specification & Semantic search interface", in particular Chapter 5 ("Named Entity Recognition for Amharic Using Deep Learning"), which describes the two versions of the Amharic Named Entity Recognizer together with some preliminary evalution of them. The other chapters of that deliverable give the reasoning for the setup and report on some related experiments. The present text will thus only describe the deep learning approach to named entity recognition (NER) for Amharic in some more detail and present the final evaluation of the system.

As reported in D1.1.4 & D5.2, experiments were carried out on named entity recognition both on English social media texts (tweets) and on an annotated Amharic named entity corpus, and with several different methods: rather than restricting to word space models, we used both a supervised method, namely Conditional Random Fields (CRFs), an evolutionary feature optimization approach (Differential Evolution), and a deep learning approach in the form of a recurrent neural network (bi-directional long short term memory, bi-LSTM).

2 Amharic Named Entity Corpus

The present NER system utilizes Amharic datasets annotated within the SAY project at New Mexico State University's Computing Research Laboratory that use a richer 6-class annotation scheme, with the categories person, location, organization, time, title and other (not named entity). The SAY Amharic annotations are available in 322 XML files from the Lexical Data Repository of the Ge'ez Frontier Foundation.[1] These files were here split into ten parts for ten-fold cross-validation. Table 1 shows the statistics of the training data for each fold (i.e., the sum of the other nine folds) in terms of number of sentences, tokens, and named entities, as well as the same information for the test data (i.e., each fold in itself), but in addition to the total number of named entities for each fold also the number of NEs in the test set (fold) that also could be found in the training set ('match') and those that could not ('noMatch'). Hence the 'noMatch' column shows the number of NEs unique to that specific fold.

3 Bi-directional Long Short Term Memory Model

In the first version of the Amharic NER system, a bi-directional LSTM (Long Short Term Memory) model was used. The deep learning method is divided into two parts: word embedding and bi-directional LSTM. The word embeddings were generated through word2vec, which takes inputs from large corpora and generates a word vector for each word. There are two types of embeddings: continuous-bags-of-words (CBOW) and skip-gram models (Mikolov et al., 2013,?). In the CBOW architecture, the model predicts the current word from a window of surrounding context words. In the skip-gram model, it predicts the context words using the current word. The word2vec model can be trained using a softmax function (Rong, 2014) or negative sampling

[1] `https://github.com/geezorg/data/tree/master/amharic/tagged/nmsu-say`

	Training data			Test data				
			Named			Named entities		
Fold	Sentences	Tokens	entities	Sentences	Tokens	total	match	noMatch
1	3,784	99,095	5,056	453	10,581	424	236	188
2	3,801	98,293	4,894	436	11,383	586	249	337
3	3,859	99,379	4,828	378	10,297	652	313	339
4	3,743	96,282	4,878	494	13,394	602	291	311
5	3,895	100,197	5,018	342	9,479	462	200	262
6	3,862	100,963	5,027	375	8,713	453	209	244
7	3,902	100,877	5,012	335	8,799	468	218	250
8	3,730	96,620	4,788	507	13,056	692	275	417
9	3,832	98,300	4,951	405	11,376	529	215	314
10	3,725	97,078	4,868	512	12,598	612	307	305
Total				4,237	109,676	5,480		

Table 1: Training and test data statistics

(Rong, 2014). Since word2vec is a unsupervised approach where annotations are not needed, the entire Amharic corpus (without annotations) was taken as training data for a word2vec model using skip-grams and negative sampling.

The bi-directional LSTM model classifies the words following the NE prediction pipeline shown in Figure 1. The network consists of an embedding/input layer with two hidden layers. In the output layer, the softmax (Rong, 2014) function assigns the words to six categories/labels. In the input layer, the word embeddings developed using word2vec are combined with non-context features: suffix and prefix, POS, frequency and digit-check (described further in Section 5). For the suffix and prefix features, 5-dimensional word vectors are generated for each length of suffix/prefix character(s) using word2vec. The suffix and prefix lengths are set for up to four characters, so that 40 (5x8) word vectors are generated for the suffix and prefix features. In addition, one-hot vectors are generated for each of the other features: a length 2 one-hot vector for frequency, a length 2 one-hot vector for the digit-check feature, and a length 5 one-hot vector for POS (encoding the classes nouns, verbs, infinitives, copulas, and 'others').

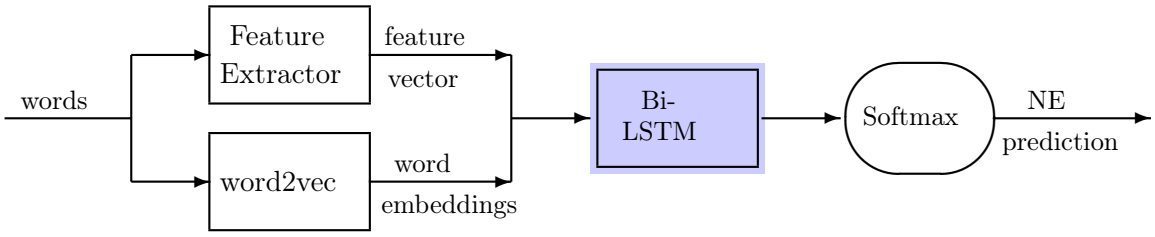

Figure 1: Baseline LSTM model

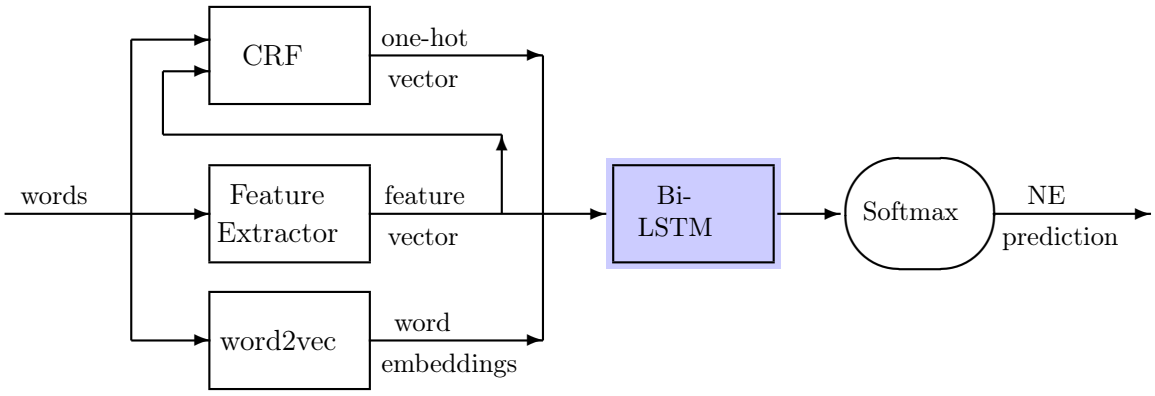

Figure 2: Pipeline of stack-based model

4 Stack-based LSTM Model

In order to optimise the system performance, a stack-based deep learner was built, combining the output of a supervised Conditional Random Field model with word embedding vectors and a set of language-independent features fed to the LSTM model to classify the words. The pipeline of the stack-based model is shown in Figure 2, with the different parts on the left of the figure working together to encode the input to the neural network:

1. A word embedding of size 300 was created from word2vec using a skip-gram model.
2. Different feature vectors of size 49 were extracted for each word in the training and test datasets, using the features described in Section 3. These feature vectors are concatenated with the word2vec output.
3. In addition, the same features were used by the CRF classifier to predict the tag for each word, with length 6 (each bit represents one class) one-hot word vectors being generated for the CRF outputs, and concatenated with the word vectors generated by the word2vec model and the feature vectors.

These three information sources (vector size 355) were then fed to a bi-directional LSTM neural network with two hidden layers in order to classify the tokens into one of the six different classes. The LSTM model was trained for 100 epochs with a batch size of 256 and with the maximum sentence length set to 70.

For comparison, another system was trained using the same setup, but without feeding feature vectors to the LSTM model. Hence that network only uses word embeddings generated by word2vec (size 300) concatenated with the CRF output vector (6 one-hot word vector), so a total input word dimension of 306.

5 Results

To establish a baseline, a supervised model was built using a CRF classifier, a supervised machine learner, built with the C^{++} based CRF^{++} package,[2] a simple, customizable, and open source implementation of CRF for segmenting or labelling sequential data. The CRF classifier was trained on the following set of features:

Local context plays an important role in identifying names. As in the work by Alemu (2013), two words before and after the focus word were used as local context (so there are four context features).

Part-of-speech tags extracted for each token using HornMorpho (Gasser, 2011), which is one of the few resources that already are available for Ethiopian languages and provides some morphological processing for Amharic, Tigrinya and Afaan Oromo.

Word suffix and prefix obtained by stripping a fixed number (up to 4) of characters from the beginning and end of the current word.

Word frequency: Less frequent words were found to often belong to named entities. If the pre-calculated frequency from the training/test data of the current word is less than a certain threshold, this binary feature is set. The thresholds were empirically set to 10 and 4 for training and test data, respectively (the frequency count threshold for the test data has to be lower than that for the training data, since there are a lot fewer instances in the test data than in the training data).

Digit check: in particular to identify the 'time' class, it is helpful to mark if a token contains any digit(s). Hence this binary flag is set for tokens containing at least one digit.

After 10-fold cross-validation, the CRF classifier achieved the average precision, recall and F_1-scores of 85.02%, 61.67% and 71.44%, respectively. For each fold, the recall, precision and F-measure values are given on the left side of Table 2.

The first version of the deep learning-based NER system was the LSTM described in Section 3, utilizing feature vectors based on the language independent features (except the context feature) that were added to the word vectors built from skip-gram word2vec model. The average recall, precision and F-scores after 10-fold cross-validation are shown on the right side of Table 2: the model using both word vectors and features achieves an average precision of 77.2% and recall of 63.4%, for a 69.7% F_1-score. It thus slightly improves the recall compared to the CRF-based classifier, but at the price of a clearly lower precision.

The performance of the stacked-based LSTM model incorporating two information sources (word vectors and CRF outputs), but not the feature vectors reached the average precision, recall and F-measure values of 85.91%, 65.33% and 74.10%, respectively, as shown on the left side of Table 3 ('No-feat LSTM'), hence surpassing both the baseline and the pure LSTM system on all accounts.

[2] `http://crfpp.sourceforge.net`

Fold	CRF classifier			word2vec + features		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score
1	0.8726	0.6887	0.7698	0.7930	0.6667	0.7244
2	0.8070	0.6197	0.7010	0.7016	0.6490	0.6743
3	0.8499	0.6044	0.7064	0.8093	0.6290	0.7079
4	0.8608	0.6399	0.7341	0.7766	0.6516	0.7087
5	0.8492	0.5580	0.6738	0.7792	0.5974	0.6763
6	0.8335	0.6039	0.7004	0.7614	0.6073	0.6757
7	0.8611	0.6072	0.7122	0.7859	0.6156	0.6904
8	0.8733	0.5796	0.6968	0.7578	0.6360	0.6916
9	0.8144	0.6133	0.6997	0.7464	0.6467	0.6930
10	0.8804	0.6530	0.7498	0.8055	0.6633	0.7275
Avg.	0.8502	0.6167	0.7144	0.7717	0.6363	0.6970

Table 2: Baseline systems: CRF classifier and LSTM with word2vec plus features

The stacked-based model incorporating all three information sources (word vectors, features and CRF outputs) out-performed all previous models, with the 10-fold cross-validation results shown in the middle of Table 3, so reaching average precision, recall and F-measure values of 85.97%, 65.51% and 74.26%, respectively, using the outputs of the previous CRF learning classifier along with the feature vectors and the word vectors from the word2vec model. However, the improvements compared to the 'No-feat' LSTM model are small, indicating that the mileage stemming from the language-independent feature set is not very significant.

Fold	No-feat LSTM			Stack-based LSTM			Voting-based Ensemble		
	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score
1	0.8716	0.7242	0.7911	0.8746	0.7267	0.7938	0.7511	0.8026	0.7760
2	0.8035	0.6661	0.7284	0.8048	0.6669	0.7293	0.7517	0.7052	0.7277
3	0.8760	0.6405	0.7400	0.8760	0.6405	0.7400	0.7787	0.7273	0.7521
4	0.8632	0.6726	0.7561	0.8643	0.6734	0.7567	0.7703	0.7356	0.7525
5	0.8750	0.5814	0.6986	0.8756	0.5847	0.7011	0.7557	0.6687	0.7096
6	0.8401	0.6338	0.7225	0.8431	0.6361	0.7251	0.7657	0.7123	0.7381
7	0.8807	0.6615	0.7555	0.8807	0.6615	0.7555	0.7295	0.7416	0.7355
8	0.8940	0.6169	0.7301	0.8917	0.6199	0.7313	0.7296	0.7024	0.7158
9	0.8195	0.6704	0.7375	0.8208	0.6722	0.7391	0.7325	0.7390	0.7357
10	0.8617	0.6656	0.7511	0.8658	0.6687	0.7546	0.7592	0.7551	0.7571
Avg.	0.8591	0.6533	0.7410	**0.8597**	0.6551	**0.7426**	0.7524	**0.7289**	0.7400

Table 3: A stack-based LSTM without feature vector input (but using word2vec and CRF outputs), a stack-based LSTM using all three information sources, and a voting-based ensemble of a CRF classifier and an LSTM model using word2vec and features.

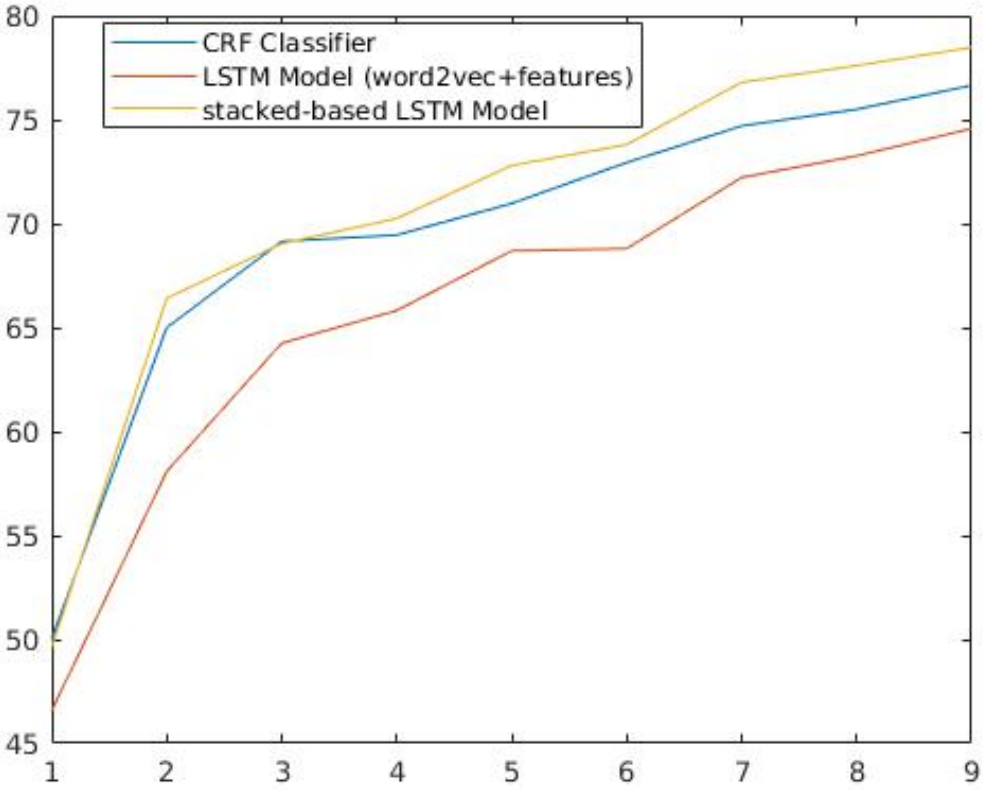

Figure 3: F-measure vs. size of training data (each step adds on average 424 sentences)

As an alternative to combining the CRF classifier and the LSTM in a stack-based model, a set of experiments were run where they were instead emsembled in a voting-based scheme. As can be seen in the right-most part of Table 3, a combination of CRF with an LSTM trained on the word2vec word embeddings and the language independent feature set performed better than the stack-based LSTM model in terms of recall, but worse than all the other models in terms of precision, for an average F-score which was slightly lower than both versions of the stacked LSTM (i.e., both with and without the features in the input set).

In order to investigate the cause of the low recall values of the stacked LSTM model, another set of experiments was run, exploring the effect of increasing the size of the training dataset. Here Figure 3 represents the variation of F-measure values with respect to different sizes of training datasets. The three graphs in the figure compare the performance of the stack-based LSTM model to that of the plain CRF classifier and the LSTM model trained on word2vec word embeddings and the feature vectors, in each case showing how the performance increases as more data (another fold) is added to the training dataset. As can be seen, all three systems improve rapidly as the first three folds are added, but (as could be expected) performance keeps on improving as more data is added, indicating that all systems would benefit from having access to even more training data.

6 Conclusion

Here we have experimented with a system for named entity recognition for Amharic, an under-resourced language. A set of language independent features was developed to extract Amharic named entities using a supervised CRF classifier and bi-directional LSTM model. Better performance was achieved after creating (unsupervised) word embeddings based on the output of the supervised model and the feature vectors together with word2vec word vectors, and then feeding the result to the neural network for training and classification. When the outputs of the CRF model were concatenated with the word embeddings, the recurrent network outperformed the other models. This may be since many of the tags/classes are identified already by the CRF model. However, the stack-based LSTM model utilizing the CRF output as an information source clearly improved on the CRF classifier itself, in particular in terms of recall. A voting-based ensemble solution using the CRF classifier and an LSTM built only on word embeddings and feature vectors showed further improvements to recall, but at the price of substantially lower precision.

Tools and resources that can help reduce language barriers and thereby provide people all over the world with improved access to information and services will have beneficial effects for most sectors of society and in the long-term contribute to the development of technology that will enable massive social and economic transformations. The present system takes a small but important step in the direction of developing such tools, but the error levels are still high and many names were not identified by the system or classified into the wrong NE categories. Notably though, one error source is that many names in the training data are annotated as non-entities, but in test data the names are annotated as named entities. However, the main cause of the low recall is most likely insufficient number of training instances, which are further reduced both by Amharic being an agglutinative language and by it lacking spelling standard for many names. This is in line with the results reported by Poostchi et al. (2016) who carried out a similar NE task on Persian, another under-resourced language, using an almost equal-sized corpus (250K tokens, of which about 10% were named entities). They compared an SVM-HMM based approach to CRF and a recurrent neural network, with the SVM-based classifier performing best, potentially since their dataset also was too small for the neural network to be trained efficiently.

In the future it would be reasonable to also develop some language dependent features to improve the performance. A set of models can also be generated by using several different classifiers and ensemble these models with the help of an evolutionary algorithm. It might also be possible to utilize the word embeddings generated for Amharic in the Polyglot project (Al-Rfou et al., 2013). They have created word embeddings for the more than 100 languages that have at least 10,000 Wikipedia entries, which mainly include European and Asian languages (and some artificial languages), but in addition to Arabic also a few Sub-Saharan African languages such as Yoruba, Swahili, Africaans and Amharic.[3] Alternatively, the cleaned Amharic web corpus created in the HaBiT project could be used as the basis for creating word embeddings.

Furthermore, the word2vec model used here was built on skip-grams that predict the context words using the current word. An alternative would be to use the continuous-bags-of-words

[3] `http://bit.ly/embeddings`

(CBOW) model, which basically does the opposite and predicts the current word from a window of surrounding context words.

An alternative to using the LSTM recurrent neural network (RNN) would be to use a Convolutional Neural Network (CNN). In general, RNNs tend to perform better for tasks where the sequential nature of human language can be exploited, but which of the two strategies work best for a specific task is often an empirical question. See Gambäck and Sikdar (2017) for a small study on using a CNN for a language classification task for Twitter data (identifying hate speech).

References

Al-Rfou, R., B. Perozzi, and S. Skiena (2013, August). Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CONLL'13)*, Sofia, Bulgaria, pp. 586–594. ACL.

Alemu, B. (2013, June). A named entity recognition for Amharic. Master's thesis, School of Information Science, Addis Ababa University, Addis Ababa, Ethiopia.

Gambäck, B. and U. K. Sikdar (2017, August). Using convolutional neural networks to classify hate-speech. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. ACL. 1st Workshop on Abusive Language Online.

Gasser, M. (2011, May). HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. In *Proceedings of Conference on Human Language Technology for Development*, Alexandria, Egypt, pp. 94–99.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. *CoRR abs/1301.3781*.

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, Red Hook, NY, USA, pp. 3111–3119. Curran Associates.

Poostchi, H., E. Z. Borzeshi, M. Abdous, and M. Piccardi (2016, December). PersoNER: Persian named-entity recognition. In *Proceedings of the 26th International Conference on Computational Linguistics*, Osaka, Japan, pp. 3381–3389. ACL.

Rong, X. (2014). word2vec parameter learning explained. *CoRR abs/1411.2738*.