Zero-shot Sequence Labeling: Transferring Knowledge from Sentences to Tokens

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Anders Søgaard
CoAStaL DIKU
Department of Computer Science
University of Copenhagen
Denmark
soegaard@di.ku.dk

Abstract

Can attention- or gradient-based visualization techniques be used to infer token-level labels for binary sequence tagging problems, using networks trained only on sentence-level labels? We construct a neural network architecture based on soft attention, train it as a binary sentence classifier and evaluate against token-level annotation on four different datasets. Inferring token labels from a network provides a method for quantitatively evaluating what the model is learning, along with generating useful feedback in assistance systems. Our results indicate that attention-based methods are able to predict token-level labels more accurately, compared to gradient-based methods, sometimes even rivaling the supervised oracle network.

1 Introduction

Sequence labeling is a structured prediction task where systems need to assign the correct label to every token in the input sequence. Many NLP tasks, including part-of-speech tagging, named entity recognition, chunking, and error detection, are often formulated as variations of sequence labeling. Recent state-of-the-art models make use of bidirectional LSTM architectures (Irsoy and Cardie, 2014), character-based representations (Lample et al., 2016), and additional external features (Peters et al., 2017). Optimization of these models requires appropriate training data where individual tokens are manually labeled, which can be time-consuming and expensive to obtain for each different task, domain and target language.

In this paper, we investigate the task of performing sequence labeling without having access to any training data with token-level annotation. Instead of training the model directly to predict the label for each token, the model is optimized using a sentence-level objective and a modified version of the attention mechanism is then used to infer labels for individual words.

While this approach is not expected to outperform a fully supervised sequence labeling method, it opens possibilities for making use of text classification datasets where collecting token-level annotation is not possible or cost-effective.

Inferring token-level labels from a text classification network also provides a method for analyzing and interpreting the model. Previous work has used attention weights to visualize the focus of neural models in the input data. However, these analyses have largely been qualitative examinations, looking at only a few examples from the datasets. By formulating the task as a zero-shot labeling problem, we can provide quantitative evaluations of what the model is learning and where it is focusing. This will allow us to measure whether the features that the model is learning actually match our intuition, provide informative feedback to end-users, and guide our development of future model architectures.

2 Network Architecture

The main system takes as input a sentence, separated into tokens, and outputs a binary prediction as the label of the sentence. We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) architecture for sentence classification, with dynamic attention over words for constructing the sentence representations. Related architectures have been successful for machine translation (Bahdanau et al., 2015), sentence summarization (Rush and Weston, 2015), entailment detection (Rocktäschel et al., 2016), and error correction (Ji et al., 2017). In this work, we modify the attention mechanism and training objective in order to make the resulting network suitable for

Proceedings of NAACL-HLT 2018, pages 293–302
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics

also inferring binary token labels, while still performing well as a sentence classifier.

Figure 1 contains a diagram of the network architecture. The tokens are first mapped to a sequence of word representations $[w_1, w_2, w_3, ..., w_N]$, which are constructed as a combination of regular word embeddings and character-based representations, following Lample et al. (2016). These word representations are given as input to a bidirectional LSTM which iteratively passes through the sentence in both directions. Hidden representations from each direction are concatenated at every token position, resulting in vectors h_i that are focused on a specific word but take into account the context on both sides of that word. We also include a transformation with $tanh$ activation, which helps map the information from both directions into a joint feature-space:

$$\overrightarrow{h_i} = LSTM(w_i, \overrightarrow{h_{i-1}}) \quad (1)$$

$$\overleftarrow{h_i} = LSTM(w_i, \overleftarrow{h_{i+1}}) \quad (2)$$

$$\widetilde{h_i} = [\overrightarrow{h_i}; \overleftarrow{h_i}] \qquad h_i = tanh(W_h \widetilde{h_i} + b_h) \quad (3)$$

where W_h is a parameter matrix and b_h is a parameter vector, optimized during training.

Next, we include an attention mechanism that allows the network to dynamically control how much each word position contributes to the combined representation. In most attention-based systems, the attention amount is calculated in reference to some external information. For example, in machine translation the attention values are found based on a representation of the output that has already been generated (Bahdanau et al., 2015); in question answering, the attention weights are calculated in reference to the input question (Hermann et al., 2015). In our task there is no external information to be used, therefore we predict the attention values directly based on h_i, by passing it through a separate feedforward layer:

$$e_i = tanh(W_e h_i + b_e) \quad (4)$$

$$\widetilde{e_i} = W_{\widetilde{e}} e_i + b_{\widetilde{e}} \quad (5)$$

where $W_{\widetilde{e}}$, $b_{\widetilde{e}}$, W_e and b_e are trainable parameters and $\widetilde{e_i}$ results in a single scalar value. This method is equivalent to calculating the attention weights in reference to a fixed weight vector, which is optimized during training. Shen and Lee (2016) proposed an architecture for dialogue act detection where the attention values are found based on a separate set of word embeddings. We found that the method described above was consistently equivalent or better in development experiments, while requiring a smaller number of parameters.

The values of $\widetilde{e_i}$ are unrestricted and should be normalized before using them for attention, to avoid sentences of different length having representations of different magnitude. The common approach is to use an exponential function to transform the value, and then normalize by the sum of all values in the sentence:

$$a_i = \frac{exp(\widetilde{e_i})}{\sum_{k=1}^{N} exp(\widetilde{e_k})} \quad (6)$$

The value a_i is now in a range $0 \leq a_i \leq 1$ and higher values indicate that the word at position i is more important for predicting the sentence class. The network learns to predict informative values for a_i based only on the sentence objective, without receiving token-level supervision. Therefore, we can use these attention values at each token in order to infer an unsupervised sequence labeling output.

The method in Equation 6 is well-suited for applications such as machine translation – the exponential function encourages the attention to prioritize only one word in the sentence, resulting in a word-word alignment. However, the same function is less suitable for our task of unsupervised sequence labeling, as there is no reason to assume that exactly one word has a positive label. An input sentence can contain more than one tagged token, or it can contain no tokens of interest, and this should be reflected in the predictions.

Instead of the exponential function, we make use of the logistic function σ for calculating soft attention weights:

$$\widetilde{a_i} = \sigma(\widetilde{e_i}) \qquad a_i = \frac{\widetilde{a_i}}{\sum_{k=1}^{N} \widetilde{a_k}} \quad (7)$$

where each $\widetilde{a_i}$ has an individual value in the range $0 \leq \widetilde{a_i} \leq 1$ and a_i is normalized to sum up to 1 over all values in the sentence. The normalized weights a_i are used for combining the context-conditioned hidden representations from Equation

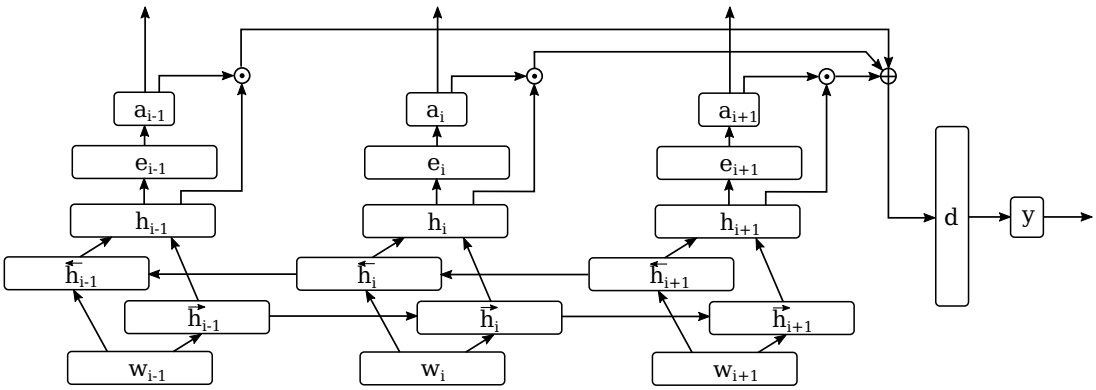

Figure 1: The neural network architecture for zero-shot sequence labeling. The soft attention values a_i are used for weighting hidden representations h_i as well as providing a binary label for each token. The network is only optimized through the sentence classification objective, predicting the sentence-level label y.

3 into a single sentence representation:

$$c = \sum_{i=1}^{N} a_i h_i \tag{8}$$

In addition, we can use the pre-normalization value $\widetilde{a_i}$ as a score for sequence labeling, with a natural decision boundary of 0.5 – higher values indicate that the token at position i is important and should be labeled positive, whereas lower values suggest the token is largely ignored for sentence classification and can receive a negative label. Attention weights with sigmoid activation have been shown to also improve performance on classification tasks (Shen and Lee, 2016), which indicates that this architecture has the benefit of being both accurate and interpretable on the token level.

Finally, we pass the sentence representation c through a feedforward layer and predict a binary label for the overall sentence:

$$d = tanh(W_d c + b_d) \tag{9}$$

$$y = \sigma(W_y d + b_y) \tag{10}$$

where d is a sentence vector and y is a single value between $0 \leq y \leq 1$, with values higher than 0.5 indicating a positive class and lower values indicating a negative prediction.

In order to optimize the model, we use several different loss functions. The first is the squared loss which optimizes the sentence-level score prediction to match the gold label in the annotation:

$$L_1 = \sum_j (y^{(j)} - \widetilde{y}^{(j)})^2 \tag{11}$$

where $y^{(j)}$ is the predicted score for the j-th sentence, and $\widetilde{y}^{(j)}$ is the true binary label $(0, 1)$ for the j-th sentence.

In addition, we want to encourage the model to learn high-quality token-level labels as part of the attention weights. While the model does not have access to token-level annotation during training, there are two constraints that we can take advantage of:

1. Only some, but not all, tokens in the sentence can have a positive label.

2. There are positive tokens in a sentence only if the overall sentence is positive.

We can then construct loss functions that encourage the model to optimize for these constraints:

$$L_2 = \sum_j (min_i(\widetilde{a_i}) - 0)^2 \tag{12}$$

$$L_3 = \sum_j (max_i(\widetilde{a_i}) - \widetilde{y}^{(j)})^2 \tag{13}$$

where $min_i(\widetilde{a_i})$ is the minimum value of all the attention weights in the sentence and $max_i(\widetilde{a_i})$ is the corresponding maximum value. Equation 12 optimizes the minimum unnormalized attention weight in a sentence to be 0, satisfying the constraint that all tokens in a sentence should not have a positive token-level label. Equation 13 then optimizes for the maximum unnormalized attention weight in a sentence to be equal to the gold label for that sentence, which is either 0 or 1, incentivizing the network to only assign large attention

weights to tokens in positive sentences. These objectives do not provide the model with additional information, but serve to push the attention scores to a range that is suitable for binary classification.

We combine all of these loss objectives together for the main optimization function:

$$L = L_1 + \gamma(L_2 + L_3) \tag{14}$$

where γ is used to control the importance of the auxiliary objectives.

3 Alternative Methods

We compare the attention-based system for inferring sequence labeling with 3 alternative methods.

3.1 Labeling Through Backpropagation

We experiment with an alternative method for inducing token-level labels, based on visualization methods using gradient analysis. Research in computer vision has shown that interpretable visualizations of convolutional networks can be obtained by analyzing the gradient after a single backpropagation pass through the network (Zeiler and Fergus, 2014). Denil et al. (2014) extended this approach to natural language processing, in order to find and visualize the most important sentences in a text. Recent work has also used the gradient-based approach for visualizing the decisions of text classification models on the token level (Li et al., 2016; Alikaniotis et al., 2016). In this section we propose an adaptation that can be used for sequence labeling tasks.

We first perform a forward pass through the network and calculate the predicted sentence-level score y. Next, we define a pseudo-label $y^* = 0$, regardless of the true label of the sentence. We then calculate the gradient of the word representation w_i with respect to the loss function using this pseudo-label:

$$g_i = \frac{\partial L_1}{\partial w_i}\Big|_{(y^*,y)} \tag{15}$$

where L_1 is the squared loss function from Equation 11. The magnitude of g_i, $|g_i|$ can now be used as an indicator of how important that word is for the positive class. The intuition behind this approach is that the magnitude of the gradient indicates which individual words need to be changed the most in order to make the overall label of the sentence negative. These are the words that are contributing most towards the positive class and should be labeled as such individually.

An obstacle in using this score for sequence labeling comes from the fact that there is no natural decision boundary between the two classes. The magnitude of the gradient is not constrained to a specific range and can vary quite a bit depending on the sentence length and the predicted sentence-level score. In order to map this magnitude to a decision, we analyze the distribution of magnitudes in a sentence. Intuitively, we want to detect outliers – scores that are larger than expected. Therefore, we map all the magnitudes in a sentence to a Gaussian distribution and set the decision boundary at 1.5 standard deviations. Any word that has a gradient magnitude higher than that will be tagged with a positive class for sequence labeling. If all the magnitudes in a sentence are very similar, none of them will cross this threshold and therefore all words will be labeled as negative.

We calculate the gradient magnitude using the same network architecture as described in Section 2, at word representation w_i after the character-based features have been included. The attention-based architecture is not necessary for this method, therefore we also report results using a more traditional bidirectional LSTM, concatenating the last hidden states from both directions and using the result as a sentence representation for the main objective.

3.2 Relative Frequency Baseline

The system for producing token-level predictions based on sentence-level training data does not necessarily need to be a neural network. As the initial experiment, we trained a Naive Bayes classifier with n-gram features on the annotated sentences and then used it to predict a label only based on a window around the target word. However, this did not produce reliable results – since the classifier is trained on full sentences, the distribution of features is very different and does not apply to a window of only a few words.

Instead, we calculate the relative frequency of a feature occurring in a positive sentence, normalized by the overall frequency of the feature, and calculate the geometric average over all features that contain a specific word:

$$r_k = \frac{c(X_k = 1, Y = 1)}{\sum_{z \in (0,1)} c(X_k = 1, Y = z)} \tag{16}$$

$$score_i = \sqrt[|F_i|]{\prod_{k \in F_i} r_k} \quad (17)$$

where $c(X_k = 1, Y = 1)$ is the number of times feature k is present in a sentence with a positive label, F_i is the set of n-gram features present in the sentence that involve the i-th word in the sentence, and $score_i$ is the token-level score for the i-th token in the sentence. We used unigram, bigram and trigram features, with extra special tokens to mark the beginning and end of a sentence.

This method will assign a high score to tokens or token sequences that appear more often in sentences which receive a positive label. While it is not able to capture long-distance context, it can memorize important keywords from the training data, such as modal verbs for uncertainty detection or common spelling errors for grammatical error detection.

3.3 Supervised Sequence Labeling

Finally, we also report the performance of a supervised sequence labeling model on the same tasks. This serves as an indicator of an upper bound for a given dataset – how well the system is able to detect relevant tokens when directly optimized for sequence labeling and provided with token-level annotation.

We construct a bidirectional LSTM tagger, following the architectures from Irsoy and Cardie (2014), Lample et al. (2016) and Rei (2017). Character-based representations are concatenated with word embeddings, passed through a bidirectional LSTM, and the hidden states from both direction are concatenated. Based on this, a probability distribution over the possible labels is predicted and the most probable label is chosen for each word. While Lample et al. (2016) used a CRF on top of the network, we exclude it here as the token-level scores coming from that network do not necessarily reflect the individual labels, since the best label sequence is chosen globally based on the combined sentence-level score. The supervised model is optimized by minimizing cross-entropy, training directly on the token-level annotation.

4 Datasets

We evaluate the performance of zero-shot sequence labeling on 3 different datasets. In each experiment, the models are trained using only sentence-level annotation and then evaluated based on token-level annotation.

4.1 CoNLL 2010 Uncertainty Detection

The CoNLL 2010 shared task (Farkas et al., 2010) investigated the detection of uncertainty in natural language texts. The use of uncertain language (also known as hedging) is a common tool in scientific writing, allowing scientists to guide research beyond the evidence without overstating what follows from their work. Vincze et al. (2008) showed that 19.44% of sentences in the biomedical papers of the BioScope corpus contain hedge cues. Automatic detection of these cues is important for downstream tasks such as information extraction and literature curation, as typically only definite information should be extracted and curated.

The dataset is annotated for both hedge cues (keywords indicating uncertainty) and scopes (the area of the sentence where the uncertainty applies). The cues are not limited to single tokens, and can also consist of several disjoint tokens (for example, *"either ... or ..."*). An example sentence from the dataset, with bold font indicating the hedge cue and curly brackets marking the scope of uncertainty:

> Although IL-1 has been reported to contribute to Th17 differentiation in mouse and man, it remains to be determined {**whether** therapeutic targeting of IL-1 will substantially affect IL-17 in RA}.

The first subtask in CoNLL 2010 was to detect any uncertainty in a sentence by predicting a binary label. The second subtask required the detection of all the individual cue tokens and the resolution of their scope. In our experiments, we train the system to detect sentence-level uncertainty, use the architecture to infer the token-level labeling and evaluate the latter on the task of detecting uncertainty cues. Since the cues are defined as keywords that indicate uncertainty, we would expect the network to detect and prioritize attention on these tokens. We use the train/test data from the second task, which contains the token-level annotation needed for evaluation, and randomly separate 10% of the training data for development.

4.2 FCE Error Detection

Error detection is the task of identifying tokens which need to be edited in order to produce a

	CoNLL 2010					FCE				
	Sent F_1	MAP	P	R	F_1	Sent F_1	MAP	P	R	F_1
Supervised	-	96.54	78.92	79.41	79.08	-	59.13	49.15	26.96	34.76
Relative freq	-	81.78	15.94	**79.98**	26.59	-	37.75	14.37	**86.36**	24.63
LSTM-LAST-BP	84.42	77.90	7.16	66.64	12.92	85.10	46.12	**29.49**	16.07	20.80
LSTM-ATTN-BP	**84.94**	80.38	9.13	71.42	16.18	**85.14**	44.52	27.62	17.81	21.65
LSTM-ATTN-SW	**84.94**	**87.86**	**77.48**	69.54	**73.26**	**85.14**	**47.79**	28.04	29.91	**28.27**

Table 1: Results for different system configurations on the CoNLL 2010 and FCE datasets. Reporting sentence-level F_1, token-level Mean Average Precision (MAP), and token-level precision/recall/F_1.

grammatically correct sentence. The task has numerous applications for writing improvement and assessment, and recent work has focused on error detection as a supervised sequence labeling task (Rei and Yannakoudakis, 2016; Kaneko et al., 2017; Rei, 2017).

Error detection can also be performed on the sentence level – detecting whether the sentence needs to be edited or not. Andersen et al. (2013) described a practical tutoring system that provides sentence-level feedback to language learners. The 2016 shared task on Automated Evaluation of Scientific Writing (Daudaravicius et al., 2016) also required participants to return binary predictions on whether the input sentence needs to be corrected.

We evaluate our system on the First Certificate in English (FCE, Yannakoudakis et al. (2011)) dataset, containing error-annotated short essays written by language learners. While the original corpus is focused on aligned corrections, Rei and Yannakoudakis (2016) converted the dataset to a sequence labeling format, which we make use of here. An example from the dataset, with bold font indicating tokens that have been annotated as incorrect given the context:

> When the show started the person who was acting **it** was not Danny Brook and **he seemed not** to be an actor.

We train the network as a sentence-level error detection system, returning a binary label and a confidence score, and also evaluate how accurately it is able to recover the locations of individual errors on the token level.

4.3 SemEval Sentiment Detection in Twitter

SemEval has been running a series of popular shared tasks on sentiment analysis in text from social media (Nakov et al., 2013; Rosenthal et al., 2014, 2015). The competitions have included various subtasks, of which we are interested in two: Task A required the polarity detection of individual phrases in a tweet, and Task B required sentiment detection of the tweet as a whole. A single tweet could contain both positive and negative phrases, regardless of its overall polarity, and was therefore separately annotated on the tweet level.

In the following example from the dataset, negative phrases are indicated with a bold font and positive phrases are marked with italics, whereas the overall sentiment of the tweet is annotated as negative:

> They may *have* a *SuperBowl* in Dallas, but Dallas **ain't winning** a SuperBowl. **Not with that** quarterback and owner. @S4NYC @RasmussenPoll

Sentiment analysis is a three-way task, as the system needs to differentiate between positive, negative and neutral sentences. Our system relies on a binary signal, therefore we convert this dataset into two binary tasks – one aims to detect positive sentiment, the other focuses on negative sentiment. We train the system as a sentiment classifier, using the tweet-level annotation, and then evaluate the system on recovering the individual positive or negative tokens. We use the train/dev/test splits of the original SemEval 2013 Twitter dataset, which contains phrase-level sentiment annotation.

5 Implementation Details

During pre-processing, tokens are lowercased while the character-level component still retains access to the capitalization information. Word embeddings were set to size 300, pre-loaded from publicly available Glove (Pennington et al., 2014)

	SemEval Negative					SemEval Positive				
	Sent F_1	MAP	P	R	F_1	Sent F_1	MAP	P	R	F_1
Supervised	-	67.70	31.79	44.66	37.02	-	67.41	36.27	50.71	42.24
Relative freq	-	44.15	17.39	15.67	16.48	-	47.64	13.39	**54.69**	21.51
LSTM-LAST-BP	53.65	43.02	8.33	28.41	12.88	70.83	49.06	17.66	35.06	23.48
LSTM-ATTN-BP	**55.83**	50.96	11.55	**31.54**	16.90	**71.26**	53.89	23.45	34.53	27.92
LSTM-ATTN-SW	**55.83**	**54.37**	**29.41**	14.40	**19.23**	**71.26**	**56.45**	**37.19**	25.96	**30.45**

Table 2: Results for different system configurations on the SemEval Twitter sentiment dataset, separated into positive and negative sentiment detection. Reporting sentence-level F_1, token-level Mean Average Precision (MAP), and token-level precision/recall/F_1.

embeddings and fine-tuned during training. Character embeddings were set to size 100. The recurrent layers in the character-level component have hidden layers of size 100; the hidden layers $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ are size 300. The hidden combined representation h_i was set to size 200, and the attention weight layer e_i was set to size 100. Parameter γ was set to 0.01 based on development experiments.

The model was implemented using Tensorflow (Abadi et al., 2016). The network weights were randomly initialized using the uniform Glorot initialization method (Glorot and Bengio, 2010) and optimization was performed using AdaDelta (Zeiler, 2012) with learning rate 1.0. Dropout (Srivastava et al., 2014) with probability 0.5 was applied to word representations w_i and the composed representations h_i after the LSTMs. The training was performed in batches of 32 sentences. Sentence-level performance was observed on the development data and the training was stopped if performance did not improve for 7 epochs. The best overall model on the development set was then used to report performance on the test data, both for sentence classification and sequence labeling. In order to avoid random outliers, we performed each experiment with 5 random seeds and report here the averaged results.

The code used for performing these experiments is made available online.[1]

6 Evaluation

Results for the experiments are presented in Tables 1 and 2. We first report the sentence-level F-measure in order to evaluate the performance on the general text classification objective. Next, we report the Mean Average Precision (MAP) at returning the active/positive tokens. This measure rewards systems that assign higher scores to positive tokens as opposed to negative ones, evaluating this as a ranking problem. It disregards a specific classification threshold and therefore provides a more fair evaluation towards systems that could be improved simply by choosing a different decision boundary. Finally, we also report token-level precision, recall and F-measure for evaluating the accuracy of this model as a sequence labeler.[2]

We report five different system configurations: **Relative freq** is the n-gram based approach described in Section 3.2. **Supervised** is the fully supervised sequence labeling system described in Section 3.3. **LSTM-LAST-BP** is using the last hidden states from the word-level LSTMs for constructing a sentence representation, and the backpropagation-based method from Section 3.1 for inducing token labels. **LSTM-ATTN-BP** is using the attention-based network architecture together with the backpropagation-based labeling method. **LSTM-ATTN-SW** is the method described in Section 2, using soft attention weights for sequence labeling and additional objectives for optimizing the network.

The method using attention weights achieves the best performance on all datasets, compared to other methods not using token-level supervision. On the CoNLL 2010 uncertainty detection dataset the system reaches 73.26% F-score, which is 93% of the supervised upper bound. The alternative methods using backpropagation and relative fre-

[1] http://www.marekrei.com/projects/mltagger

[2] The CoNLL 2010 shared task on uncertainty detection comes with an official scorer which requires additional steps and the detection of both cues and scopes, whereas the binary labels from the zero-shot systems are not directly applicable to this format. Similarly, error detection is commonly evaluated using $F_{0.5}$, which is motivated by end-user experience, but in this case we wish to specifically measure the tagging accuracy. Therefore we use the regular F_1 score as the main evaluation metric for both of these tasks.

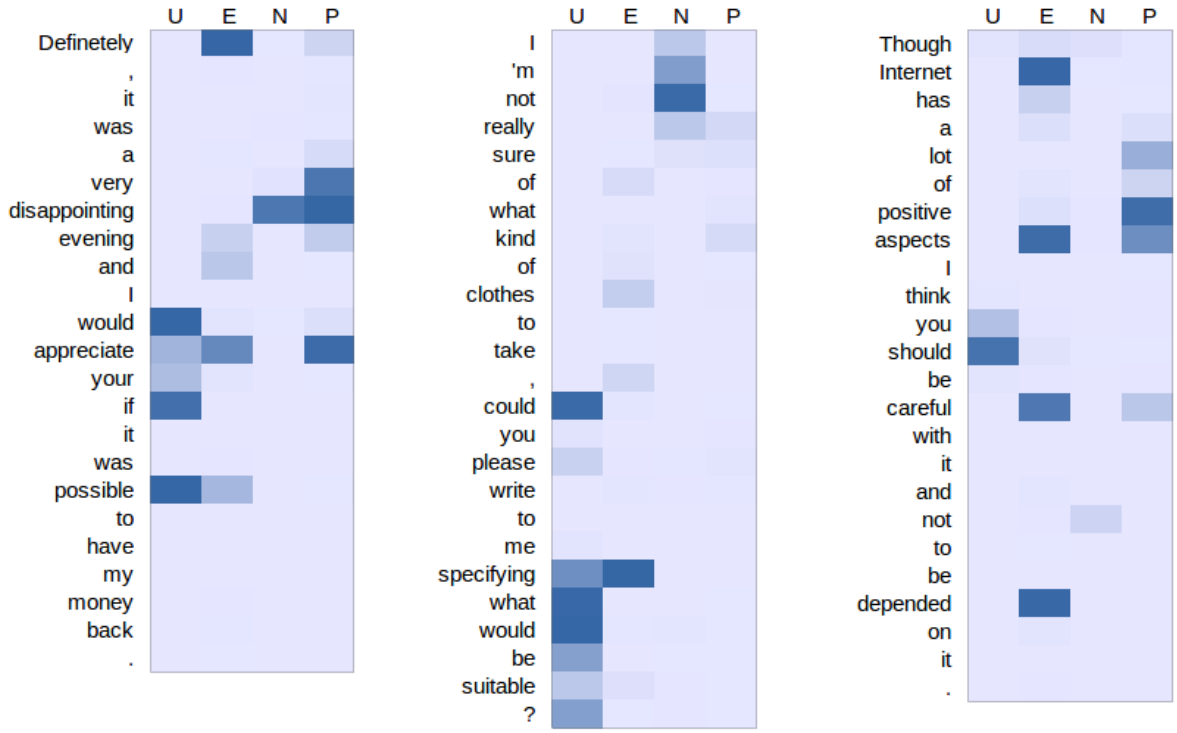

Figure 2: Example output from each of the zero-shot sequence labeling models, trained on 4 different tasks. **U**: uncertainty detection, **E**: error detection, **N**: negative sentiment detection, **P**: positive sentiment detection. Darker blue indicates higher predicted values.

quency achieve high recall values, but comparatively lower precision. On the FCE dataset, the F-score is considerably lower at 28.27% – this is due to the difficulty of the task and the supervised system also achieves only 34.76%. The attention-based system outperforms the alternatives on both of the SemEval evaluations. The task of detecting sentiment on the token level is quite difficult overall as many annotations are context-specific and require prior knowledge. For example, in order to correctly label the phrase *"have Superbowl"* as positive, the system will need to understand that organizing the Superbowl is a positive event for the city.

Performance on the sentence-level classification task is similar for the different architectures on the CoNLL 2010 and FCE datasets, whereas the composition method based on attention obtains an advantage on the SemEval datasets. Since the latter architecture achieves competitive performance and also allows for attention-based token labeling, it appears to be the better choice. Analysis of the token-level MAP scores shows that the attention-based sequence labeling model achieves the best performance even when ignoring classification thresholds and evaluating the task through ranking.

Figure 2 contains example outputs from the attention-based models, trained on each of the four datasets. In the first example, the uncertainty detector correctly picks up *"would appreciate if"* and *"possible"*, and the error detection model focuses most on the misspelling *"Definetely"*. Both the positive and negative sentiment models have assigned a high weight to the word *"disappointing"*, which is something we observed in other examples as well. The system will learn to focus on phrases that help it detect positive sentiment, but the presence of negative sentiment provides implicit evidence that the overall label is likely not positive. This is a by-product of the 3-way classification task and future work could investigate methods for extending zero-shot classification to better match this requirement.

In the second example, the system correctly labels the phrase *"what would be suitable?"* as uncertain, and part of the phrase *"I'm not really sure"* as negative. It also labels *"specifying"* as an error, possibly expecting a comma before it. In the third example, the error detection model labels *"Internet"* for the missing determiner, but also captures a more difficult error in *"depended"*,

which is an incorrect form of the word given the context.

7 Conclusion

We investigated the task of performing sequence labeling without having access to any training data with token-level annotation. The proposed model is optimized as a sentence classifier and an attention mechanism is used for both composing the sentence representations and inferring individual token labels. Several alternative models were compared on three tasks – uncertainty detection, error detection and sentiment detection.

Experiments showed that the zero-shot labeling system based on attention weights achieved the best performance on all tasks. The model is able to automatically focus on the most salient areas of the sentence, and additional objective functions along with the soft attention mechanism encourage it to also perform well as a sequence labeler. The zero-shot labeling task can provide a quantitative evaluation of what the model is learning, along with offering a low-cost method for creating sequence labelers for new tasks, domains and languages.

Acknowledgments

We would like to thank the NVIDIA Corporation for the donation of the Titan GPU that was used for this research. Anders Søgaard was partially funded by the ERC Starting Grant LOWLANDS No. 313695.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Arxiv preprint arXiv:1603.04467* https://doi.org/10.1038/nn.3331.

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic Text Scoring Using Neural Networks. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* .

Øistein E. Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications* http://www.aclweb.org/anthology/W13-1704.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*. https://doi.org/10.1146/annurev.neuro.26.041002.131047.

Vidas Daudaravicius, Rafael E Banchs, Elena Volodina, and Courtney Napoles. 2016. A Report on the Automatic Evaluation of Scientific Writing Shared Task. *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications* pages 53–62.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network. In *Arxiv preprint arXiv:1406.3830*. http://arxiv.org/abs/1406.3830.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 shared task: learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, July, pages 1–12. http://dl.acm.org/citation.cfm?id=1870535.1870536.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)* 9:249–256. https://doi.org/10.1.1.207.2059.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems (NIPS 2015)*. pages 1–14. https://doi.org/10.1109/72.410363.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation* 9. https://doi.org/10.1.1.56.7752.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *ACL 2017*. pages 753–762. https://doi.org/10.18653/v1/P17-1070.

Masahiro Kaneko, Yuya Sakaizawa, and Mamoru Komachi. 2017. Grammatical Error Detection Using Error- and Grammaticality-Specific Word Embeddings. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT 2016*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and Understanding Neural Models in NLP. *Proceedings of NAACL-HLT 2016* https://doi.org/10.18653/v1/N16-1082.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 task 2: Sentiment analysis in twitter. *Proceedings of the International Workshop on Semantic Evaluation, SemEval* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe : Global Vectors for Word Representation. In *Proceedings of the Empiricial Meth- ods in Natural Language Processing (EMNLP 2014)*. https://doi.org/10.3115/v1/D14-1162.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL 2017*. https://doi.org/10.18653/v1/P17-1161.

Marek Rei. 2017. Semi-supervised Multitask Learning for Sequence Labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL-2017)*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional Sequence Labeling Models for Error Detection in Learner Writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. https://aclweb.org/anthology/P/P16/P16-1112.pdf.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, and Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. *International Conference on Learning Representations* .

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* http://alt.qcri.org/semeval2015/cdrom/pdf/SemEval078.pdf.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* http://alt.qcri.org/semeval2014/cdrom/pdf/SemEval009.pdf.

Alexander M Rush and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of EMNLP 2015*.

Sheng Syun Shen and Hung Yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* 08-12-September-2016:2716–2720. https://doi.org/10.21437/Interspeech.2016-1359.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)* 15. https://doi.org/10.1214/12-AOS1000.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics* 9(Suppl 11):S9. https://doi.org/10.1186/1471-2105-9-S11-S9.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. http://www.aclweb.org/anthology/P11-1019.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* http://arxiv.org/abs/1212.5701.

Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. *Computer VisionECCV 2014* 8689. https://doi.org/10.1007/978-3-319-10590-1_53.