INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

CRIM's Speech Transcription and Call Sign Detection System for the ATC Airbus Challenge task

Vishwa Gupta, Lise Rebout, Gilles Boulianne, Pierre-André Ménard, Jahangir Alam

Centre de recherche informatique de Montréal (CRIM)

{Vishwa.Gupta, Lise.rebout, Gilles.Boulianne, Pierre-Andre.Menard, Jahangir.Alam}@crim.ca

Abstract

The Airbus air traffic control challenge evaluates speech recognition and call sign detection using real conversations between air traffic controllers and pilots at Toulouse airport in France. CRIM's main contribution in acoustic modeling for transcribing these conversations is experimentation with bidirectional LSTM (BLSTM) models and lattice-free MMI (LF-MMI) trained TDNN models. Adapting these acoustic models trained from a large dataset to 40 hours of ATC acoustic training data reduces WER significantly compared to training them with the ATC data only. Multiple iterations of adaptation reduce WER for the BLSTM acoustic models significantly, but only marginally for the LF-MMI TDNN acoustic models. Constrained dialog between the air traffic controller and the pilot leads to language model perplexity below 12, and WER for leaderboard and evaluation sets of 9.98% and 9.41% respectively.

For call sign detection from the decoded transcript, we use a bidirectional LSTM followed by conditional random field classifier. This DNN architecture worked better than a finite state transducer based call sign detection. Taking a majority vote over call signs from multiple decodes reduced the call sign errors. The best F1 for call sign detection for leaderboard was 0.8289 and for evaluation 0.8017. Overall, we came 3rd in this evaluation.

Index Terms: Deep Neural Networks, DNN, Bi-directional LSTM, TDNN, air traffic control automation, call sign detection.

1. Introduction

The air traffic control (ATC) challenge was launched by Airbus in April 2018 [1]. The idea behind this challenge was to see if automated transcription of the conversations between the air traffic controllers and the pilots can reduce workload of both the air traffic controllers and the pilots. These conversations include call sign of the aircraft. An example call sign is "Airbus four four india lima". Air traffic controllers use the call sign to identify the aircraft for their instructions, while pilots use the call sign to identify their aircraft and the action they are taking. Some air traffic control announcements are directed to all the flights and contain no call signs. This challenge is for automatically transcribing the conversations and identifying the call sign in the conversation if it exists. Decoding the commands or actions from the transcriptions was not part of the challenge.

In order to facilitate acoustic and language model (LM) training, Airbus and IRIT (Institut de Recherche en Informatique de Toulouse) together recorded and transcribed around 50 hours of live air traffic conversations at Toulouse airport. In April 2018, they released 40 hours of transcribed audio for training the acoustic and language models. The challenge rules allow any external data together with this data to be used for improving the automated transcription and call sign detection task. The leaderboard data was released in mid June without transcripts. We could upload up to 2 leaderboard transcripts per day to be scored and the ranks posted on the leaderboard by IRIT. The evaluation data (5 hours) was released on July 2, 2018 and the final ranking was based on this data only. Airbus has not released the transcripts for either the leaderboard or the evaluation data, so we cannot run any further experiments to improve the systems. Our results on leaderboard/evaluation data are based on scoring done at IRIT.

For the MGB3 challenge, we used bidirectional LSTM (BLSTM) [2] and lattice-free MMI (LF-MMI) trained TDNN acoustic models [3] to get good word error rate (WER) for the BBC shows [4]. So we trained these models with all the English data available to us (over 4,000 hours), and used these models to adapt to the 40 hours of Airbus Challenge training data. The adapted models gave significantly lower WER than training these models from scratch with just the 40 hours of Airbus Challenge training data (2.7% absolute drop in WER). Multiple iterations of adaptation also reduce WER (significantly more for the BLSTM models than for the LF-MMI trained TDNN models). Combining multiple decoded transcripts with ROVER [5] reduced WER significantly.

For language modeling, we trained 3-gram, 4-gram, 5-gram and RNNLM based language models from the training transcripts only. The perplexity of these models was very low for a dev set, showing that the conversations between the air traffic controllers and the pilots are very focussed. Our best result for the evaluation set was 9.41% word error rate (WER).

For call sign detection, we tried two different algorithms: one based on a finite state transducer (FST) and another based on a DNN followed by a conditional random field (CRF) classifier. For training the FST, each training utterance was marked with the start and end of call sign. Post processing the recognized transcript with this FST gives the beginning and end of the call sign in this transcript.

For our DNN-based call sign detection, we treated call sign detection as named entity detection [6] [7] [8]. Our call sign detection was based on an existing system that searches for named entities [8] in the text. There are two bidirectional LSTMs with inputs corresponding to word embeddings and character embeddings. The pretrained word embeddings use 200-dimensional GloVe vectors [9], while the character-level embedding for the word is trained using a bi-directional LSTM with one-hot vector input for each character. The output of the last hidden layer is then input to the CRF classifier. This call sign detection system gave significantly better results than the FST based call sign detection. Taking a majority vote over call signs from multiple decodes improved the call sign detection even further. Our best result for call sign detection on the eval set was an F1 measure [1] of 0.8017.

http://dx.doi.org/10.21437/Interspeech.2019-1131

2. Training data

The 40 hours of acoustic training data provided by the organisers contains 28045 speech turns. The leaderboard is 3506 speech turns (approx. 5 hours) and the evaluation set is 3595 speech turns (approx. 5 hours). All the data is in chronological order, so each conversation is a sequence of speech turns. However, we were not provided with this information, so we assumed each speech turn to be random, and we did not exploit this sequence information. There is a call sign for each speech turn in the training set. We took 3% random utterances from the training set to create a development set, and the remaining utterances were used for training the acoustic and language models. We removed 57 utterances with a blank transcript. The resulting training set had 27149 turns and the dev set had 839 turns. The 28045 training examples contain 20438 call signs divided into 2341 distinct call signs. The vocabulary size is small (2489 distinct words and a total of 441,587 words), quite different from more conventional corpora of comparable size, for example the Twitter corpus distributed with NLTK[1] (23,164 distinct words from a total of 450,223 words).

To give some feeling for the call signs, this training set has 185 distinct aircraft codes with one to three capitalized words (for example *Air France*, *Twinjet*, ...). The call sign length varies between 1 and 13 words, with 96% between 4 to 6 words long. Short call signs, and call signs not beginning with an aircraft code (414 such examples) are due to air traffic controllers abbreviating call signs of already identified aircrafts. Noisy audio and push to talk also leads to truncated call signs. Only 96 utterances contain just a call sign, 8965 (44 %) are at the beginning, 6849 (33.5 %) at the end, 4528 (22 %) in the middle of a call.

3. Language Models

We trained the language models from the training set with 27149 turns (409k words of text). We trained 3-gram, 4-gram, 5-gram and RNNLM language models. For RNNLM, we used Mikolov's toolkit [10] to train RNNLM with 200 classes and a hidden layer of size 300. The perplexity of the various language models on the dev set with 839 turns is shown in Table 1.

Table 1: *Perplexity of the various LMs on the dev set.*

language model	Perplexity
3-gram	11.5
4-gram	9.9
5-gram	9.4
RNNLM	7.7

4. CRIM's Acoustic Models

4.1. Single decoding processes

For Airbus Challenge, we experimented with two different acoustic models: bidirectional LSTMs [2] and LF-MMI trained TDNN models [3]. We trained 3-level bidirectional LSTM acoustic models (with cell dim of 1024, hidden layer dim of 1024, and recurrent and non-recurrent projection layer dim of 128, 17.7 million parameters) from this training data. The TDNN chain models have 7 hidden layers, use ReLU of size 725, and the splice indexes are: "-1,0,1 -1,0,1,2 -3,0,3 -3,0,3 -3,0,3 -6,-3,0 0" (17.4 million parameters). All the models use 40-dimensional mfcc features together with 100 dimensional i-vectors [11] [12] [13].

We had already trained LSTM and TDNN models from a very large data set containing Hub4, RT03, RT04, Market, WSJ, Libri-Speech, switchboard, and Fisher data using the Kaldi toolkit [14]. These datasets can be obtained from LDC. We adapted these models to the ATC training data. By adaptation, we mean, the training starts with these already trained models as initial models and we carry out 6 more epochs with just the ATC training data with new alignments. The idea was to start with an acoustic model with well-trained phoneme set, and then to adapt these phonemes to the small ATC training set. The phonemes that are well represented in the ATC training audio will get trained by this ATC data, while other phonemes will still have somewhat decent representation in these models. We also trained the TDNN models from scratch just from the ATC data alone in order to see the difference in WER with/without adaptation. Table 2 compares the WER for the various systems on the 839 turn dev set.

Table 2: *WER on the 839 turn dev set for various models.*

LF-MMI TDNN	WER	BLSTM	WER
original (tdnn1)	34.6%	original (blstm1)	41.4%
from scratch	16.2%	adapt (blstm2)	10.7%
adapt from tdnn1	13.5%	readapt (blstm3)	9.4%
+sMBR	11.7%	blstm3 + 5-g LM	9.36%
+ adapt 2nd time	11.35%	+ adapt to leaderboard	9.3%
		+ RNNLM	9.06%
		+ LMs adapted to leaderboard	9.03%
		augmented data BLSTM	9.4%

Both the BLSTM and TDNN models trained on 4,000 hours of audio (without the ATC data) fare poorly with WER of 41.4% and 34.6% on the dev set (line 2, Table 2). When we train the TDNN acoustic models from scratch, we get 16.2% WER (line 3), while adapting the TDNN models with the ATC data leads to 13.5% WER (line 4). Adaptation leads to 2.7% absolute reduction in WER. Discriminative sMBR training of the TDNN models leads to 11.7% WER. Adapting 2nd time these TDNN models reduces WER to 11.35%. Adapting the BLSTM acoustic models leads to 10.7% WER, and adapting 2nd time gives 9.4% WER. Looks like after adaptation, there is significant difference in WER between BLSTM and LF-MMI trained TDNN models (9.4% vs 11.35%). Adapting to training set + leaderboard data (with recognized transcript) leads to only a small reduction in WER (line 6).

4.2. Noise Robust Acoustic Features

We tried many robust features including RMCC features (regularized MVDR cepstral coefficients) [15, 16]. The RMCC features gave lower WER than the MFCC features in Chime4 challenge [17]. However, for ATC dev set, these features gave higher WER to the extent that combining the results after ROVER [5] made WER significantly worse. The reason could be that we did not have time to train models with these features from 4000 hours of audio. We had to train the acoustic models from scratch just from the 40 hours of ATC training data.

What worked was data augmentation by adding different noises to the training set. We added aircraft noise, babble noise,

[1] http://www.nltk.org/nltk_data/

music, environment noise, and reverberation to the 40 hours of training audio. In other words, we increased the training set by a factor of 5. We adapted the original LSTM models (blstm1) to this augmented training data. This LSTM trained from augmented data gave 9.4% WER on the dev set (see last line of Table2). After ROVER, these models reduced the WER both on the dev set and on the leaderboard (see Table 3).

4.3. Submission details

For the leaderboard data, we tried many system combinations. Our initial submission for leaderboard was the ROVER [5] of 3 systems: adapted BLSTM from training set (blstm3), blstm adapted to augmented data, adapted LF-MMI trained TDNN. In each case the decoding was done using a trigram LM, rescored with 5-gram LM and the resulting lattice rescored with N-best rescoring (N=200) using the RNNLM trained from the training set (line 2, Table 3). When we add the leaderboard data (transcribed with the ROVER output) and then adapt the BLSTM models to this augmented training set, we get WER of 9.99% (line 3, Table 3). When we adapt the resulting acoustic model to just the leaderboard data with a small learning rate and only 1 epoch, we get 9.98% WER (line 4, Table 3). This is the best we could do for the leaderboard transcription. Combining with ROVER the additional transcripts only made the results a little bit worse for the leaderboard data.

Table 3: *WER on Dev, Leaderboard and Eval sets for our various submissions.*

System	Dev	Leaderboard	Eval
blstm3+ augmented-data blstm + tdnn	9.02%	10.0%	
+ leaderboard in training		9.99%	
+ adapt on leaderboard		9.98%	
ROVER of 6 systems	8.45%		9.41%

However, when we combine the different transcripts for the dev set with ROVER, the WER goes down to 8.45% (line 5, Table 3). So we used the same 6 decodes for the eval set as our first submission. The six decodes are:

1. BLSTM readapted, rescored with 5-gram LM and RNNLM.
2. augmented-data BLSTM adapted 2nd time, rescored with 5-gram LM and RNNLM.
3. LF-MMI trained TDNN adapted 2nd time, rescored with 5-gram LM and RNNLM.
4. BLSTM adapted with training + leaderboard data, rescored with 5-gram LM and RNNLM trained with training and leaderboard transcripts.
5. LF-MMI trained TDNN adapted with training + leaderboard data, rescored with 5-gram LM and RNNLM trained with training and leaderboard transcripts.
6. augmented-data BLSTM adapted with training + leaderboard data, rescored with 5-g LM and RNNLM.

The three different submissions for the Evaluation set are: first is ROVER of 6 decodes outlined above (line 5 of Table 3), second is where we add the eval set data to the acoustic training data and adapt the models one more time, and the third is where we adapt only to the Evaluation set with a small learning rate. We do not know the WER for each submission, but for our best

clear^{O} takeoff^{O} three^{O} two^{O} right^{O} **Easy^{B} two^{I} six^{I} one^{I} quebec^{I}**

Figure 1: *Example of a training instance tagged with the IOB tag set*

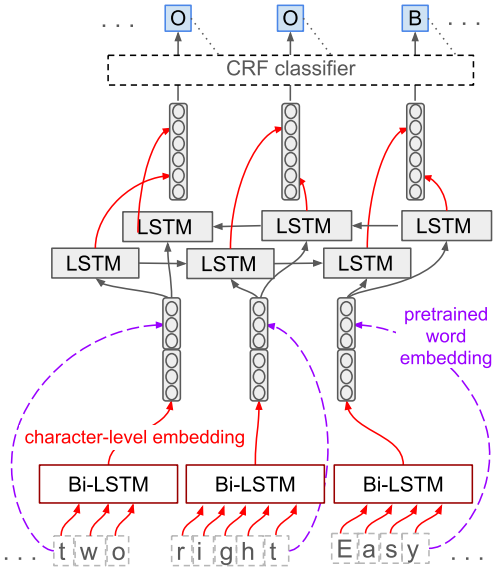

Figure 2: *Bi-LSTM - CRF architecture.*

submission, the WER was 9.41%. These submitted systems are not real-time. But the TDNN systems without RNN rescoring are real-time. So for real-time TDNN system, we lose 3.2% absolute in WER (from 8.45% to 11.68%) for the dev set.

5. Call sign detection

For call sign detection, we tried two different algorithms: one based on a finite state transducer (FST) and another based on an RNN followed by a conditional random field (CRF) classifier. To create the FST in the first algorithm, start and end markers were added before and after each call sign in the training utterances, and an N-gram language model was trained on the resulting text. The N-gram LM was converted to a weighted FST, and call sign markers were replaced with epsilon (null) symbols on the input side while kept on the output side. The recognized transcript was then composed with this FST on the right, and searched for the best path, yielding an output sequence with call sign markers at locations that maximized LM likelihood. The best result (F1 score [1] of 0.7957) was obtained with an FST corresponding to a 5-gram LM and composed with another FST to constrain start and end markers to occur in pairs.

In the second algorithm, we addressed the problem of call sign detection as a typical sequence labeling problem: we preprocessed the training transcripts to use the Inside-Outside-Beginning (IOB) tag set as shown in Figure 1. Each word in the training transcripts gets one of three posterior labels: I (inside call sign), O (outside call sign) or B (begin of call sign). The DNN architecture of our call sign detector (as shown in Figure 2) is similar to the one described in [8]: a deep neural network used as a *feature extractor* and a *tagger (or classifier)* that predicts the label (I, O or B) of each word in the sentence.

5.1. Feature extractor

The feature extractor consists of two bidirectional long short-term memory (bi-LSTM) as proposed in [20]. The first bi-LSTM is trained at character level to model sub-word information such as case or morphology. (This bi-LSTM with character input is important as around 29% of the words in the ATC training set are not part of GloVe's vocabulary.) Each character, represented by a 50-dimensional one-hot vector, is input to a fully connected layer. For each word, the final hidden states of the first bi-LSTM (25 dimensional forward and backward hidden layers) are concatenated to the 200-dimensional pre-trained GloVe word embeddings. The second bi-LSTM takes this 250-dimensional vector as input and models words and their context.

The output of the second bi-LSTM is a 200-dimensional vector from the last hidden layers (100 for the forward and 100 for the backward), and contains information about the word itself, and the word's left and right contexts. These vectors are then used as inputs to the CRF tagger.

5.2. Tagger / Classifier

We train a conditional random field (CRF) tagger (or classifier) as described in [19]. CRF is a traditional algorithm to tag sequences of units. Our linear-chain CRF considers not only the unit and its context but also the previous tag. Rather than using manually defined features, we used the output vectors of the previous bi-LSTM as features as proposed in [20]. Our loss function is the sum of the Negative Log-Likelihood (NLL) of the correct tag for each word in the sentence. The Viterbi algorithm is used for decoding.

5.3. Training and results

We trained the feature extractor and the classifier simultaneously using PyTorch[2]. We set apart 5% of the training set as validation set to do early stopping. To avoid overfitting, we applied dropout between the two bi-LSTM with a drop-out probability of 0.25. The Adam optimizer was used for gradient descent with gradient clipping for regularization.

Each audio file goes through multiple decodes resulting in multiple transcripts. For each of these transcripts, we extract a call sign through this DNN-based call sign extraction system. We then take a majority vote of the extracted call signs. The most frequent call sign is then our final call sign for that audio. This majority vote improved the call sign detection system significantly. The F1 measure for leaderboard call sign detection for different algorithms are shown in Table 4.

Table 4: *Comparison of call sign detection algorithms on the leaderboard dataset*

Algorithm	call sign F1
FST	0.7957
LSTM-CRF	0.8207
LSTM-CRF + majority vote	0.8289
LSTM-CRF + majority vote + postprocessing	0.8021

5.4. Post-processing

After a qualitative review of the call signs detected by the above classifier, we decided to implement a pattern-based post-processing step. This step has two goals : filter out wrong call signs and find new call signs that may have been missed.

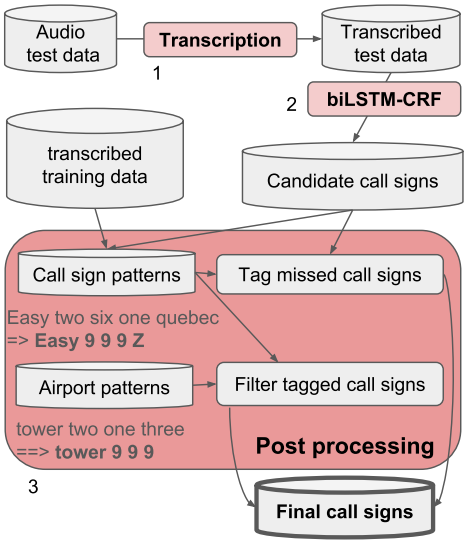

Figure 3: *Call sign enrichment and filtering by post-processing*

The airline patterns were computed for each airline by replacing each number by a 9 and each letter of the NATO phonetic alphabet by an A. Placeholders were introduced to simulate hesitations. Both the textual training data and the candidate call signs were used for the extraction of patterns. We also define airport or position patterns that can easily be mixed up with airline call signs. Such patterns begin with terms like *tower* or *altitude*. See Figure 3 for a visual description of our system. Applying post processing on the call signs detected in the leaderboard dataset gave us a worse F1 score than our previous results (0.8021 instead of 0.8289). However the new call signs detected seemed to be valid call signs. Also, leaderboard and training set call sign annotations were not always accurate. Only the evaluation set call signs were accurate. For that reason, for the evaluation set, we submitted two call sign detections based on LSTM-CRF + majority vote only, and one after this post-processing system.

6. Conclusion

The best result we obtained for the eval set was 9.41% transcription WER and F1 measure of 0.8017 for call sign detection. These results are quite good considering the various speech distortions due to transmission channels, bandwidth limitations, cockpit noises and various accents. A number of improvements have lead to these results. We found that adapting acoustic models trained from over 4,000 hours of English audio to 40 hours of ATC training data gave 2.7% absolute reduction in WER (compared to training from ATC data only). Repeated adaptation of the acoustic models to the ATC training data also gave significant reduction in WER. Adapted BLSTM acoustic models gave lower WER than adapted LF-MMI trained TDNN models. Combining transcripts from multiple decodes using ROVER also reduced WER significantly.

Call sign detection using RNN-CRF models gave better results than using finite state transducers marked with beginning and end of call signs.Taking majority vote over call signs from multiple decodes improved F1 score. Overall, we ranked third in the evaluation.

[2]https://pytorch.org/

7. References

[1] J. Farinas and T. Pellegrini, "2018 AIRBUS Air Traffic Control Challenge", online https://www.irit.fr/recherches/SAMOVA/assets/files/Seminaires/AIRBUS_ATC_Challenge_2018/Presentations/slides_ATC_workshop_4oct18_JF_TP.pdf .

[2] A. Graves, N. Jaitly, A. Mohamed, "Hybrid Speech Recognition with Deep Bidirectional LSTM", In Proc. ASRU 2013, Olomouc, Czech Republic.

[3] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, S. Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI", In Proc. Interspeech 2016, pp. 2751-2755.

[4] V. Gupta and G. Boulianne, "CRIM's system for the MGB-3 English Multi-Genre Broadcast Media Transcription", In Proc. Interspeech 2018, pp. 2653–2657.

[5] J. Fiscus, "A Post-processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)", In Proc. IEEE Workshop on Automatic Speech Recognition and Understanding, 1997, pages 347–354, Santa Barbara, CA, USA.

[6] S. Ghannay, A. Caubrière, Y. Estève, A. Laurent, E. Morin, "End-to-end named entity extraction from speech", In arXiv:1805.12045v1 [cs.CL].

[7] M. Xu, H. Jiang, S. Watcharawittayakul, "A Local Detection Approach for Named Entity Recognition and Mention Detection", In Proc. 55th Annual Meeting Assoc. for Computational Linguistics, pages 1237–1247, Vancouver, Canada, 2017.

[8] G. Bernier-Colborne, C. Barriere, P. A. Ménard, "CRIM's systems for the tri-lingual entity detection and linking task", In Proc. TAC 2017.

[9] J. Pennington, R. Socher, C. Manning, "Glove: Global vectors for word representation", inProceedings of the 2014 conference on empirical methods in natural language processing, pp. 1532–1543.

[10] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, Strategies for training large scale neural network language models, In Proc. ASRU 2011, pp. 196–201.

[11] V. Gupta, P. Kenny, P. Ouellet, T. Stafylakis, "I-vector-based speaker adaptation of deep neural networks for French broadcast audio transcription", In Proc. ICASSP 2014, Florence, Italy, pp. 6334-6338.

[12] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors", In Proc. ASRU 2013, pp. 55–59.

[13] A. Senior, I. Moreno, "Improving DNN speaker independence with i-vector inputs", In Proc. ICASSP 2014,

[14] D. Povey et. al., "The Kaldi Speech Recognition Toolkit", In Proc. ASRU 2011.

[15] M. J. Alam, P. Kenny, D. O'Shaughnessy, "Regularized Minimum Variance Distortionless Response-Based Cepstral Features for Robust Continuous Speech Recognition", Speech Communication (2015), vol. 73, pp. 28–46.

[16] M. J. Alam, P. Kenny, P, Dumouchel, D. O'Shaughnessy, "Robust Feature Extractors for Continuous Speech Recognition", In Proc. EUSIPCO (2014), Lisbon, Portugal, pp. 944–948.

[17] M. J. Alam, V. Gupta, P. Kenny, "CRIMs Speech Recognition System for the 4th CHiME Challenge", in Proc. CHiME 2016, pp. 63–67.

[18] A. Graves, A. Mohamed, and G. Hinton. "Speech Recognition with Deep Recurrent Neural Networks", In Proc. ICASSP 2013, pp. 6645–6649.

[19] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", In Proc. ICML, 2001.

[20] Z. Huang, W. Xu and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging". arXiv preprint arXiv:1508.01991.2015.