Reading Assignment 1
Hamza Rashid, 260971031
COMP 550, Fall 2024

The paper "Neural Architectures for Named Entity Recognition" by Lample et al. (2016), addressed the reliance of state-of-the-art Named Entity Recognition (NER) systems on hand-crafted features and domain-specific knowledge, a common approach at the time due to the small, supervised training corpora that were available. The authors proposed two neural architectures designed to generalize under the limited training corpora: a bidirectional-LSTM encoder pipelining input sequences to a CRF layer, and a greedy, transition-based chunking algorithm utilizing a Stack-LSTM to manage states. A key component of their methodology was the formation of character-sensitive word embeddings, to capture orthographical and morphological details. While the proposed models achieved state-of-the-art performance, some aspects of the methodology limit the scope for generalization.

The Bi-LSTM encoder consists of a forward and backward LSTM, reading the input sequence in those orders. The resulting left and right representations are then concatenated into a single vector, the word-in-context representation. The bi-directional architecture is critical for making informative encodings, as the left and right encodings may be insufficient, or capture the wrong *context*, if considered in isolation (homographical richness is an intuitive case). Furthermore, the CRF layer utilizes the formulation discussed in class, where the feature sum is taken over the state-transition and observation-emission (given by the Bi-LSTM) scores. The linear-chain CRF's ability to capture the bi-directional relationships between output labels makes it effective in addressing two problems that are closely related to classification accuracy: the strong dependence between output labels (inherent to the NER task), and, the formal grammar induced by the `IOBES (Inside, Outside, Beginning, End, Single)` tagging scheme utilized by the authors (e.g., `I-PER` cannot follow `B-LOC`).

The Transition-Based Chunking architecture consists of two stacks (an output and processing stack), a buffer (unprocessed portion of the input sequence), and three operators (`SHIFT`: moves token from buffer to stack, `OUT`: moves token from buffer to output stack, `REDUCE(y)`: pops all items from top of the stack, creating a "chunk", assigns it the label `y`, and pushes an embedding onto the output). The model concatenates embeddings of the stack, buffer, output, and operation history to maintain the full algorithm state. This embedding is used to define a probability distribution over actions at each time step, and the model is trained to maximize the conditional probability of sequences of reference actions (extracted from a labeled training corpus) given the input sentences, greedily choosing the maximum probability operator at each time step.

In designing a language agnostic approach, the authors trained their models on datasets containing English, Spanish, German, and Dutch documents. Many aspects of their methodology deal with generalizability, and preventing overfitting. This is demonstrated in their use of character-level word embeddings, dropout training, and Stochastic Gradient Descent. Character-level embeddings helped capture orthographic and morphological features indicating whether a token (or token sequence) represents a name, as well as retain meaningful data even when dealing with OOV items. The authors utilized a Bi-LSTM to process each token as a character-sequence, with the final embedding given by the concatenation of the left and right contexts, which is then concatenated with a pretrained (skip-n-gram) word embedding in a lookup table. This combination of embeddings did not enhance performance until it was supplemented with dropout training, and the authors found that the optimal setting for this parameter varied across the languages. During testing, words without an embedding in the lookup table are mapped to an `UNK` embedding, and such singletons are assigned a probability of 0.5 during training. This approach differs from those discussed in class, which either identify `OOV` words seen during testing with those that occurred with low frequency during training (e.g., frequency threshold to determine `UNK` mapping), or, deal with `OOV` items at the hyperparameter-tuning level (e.g., the Naive Bayes smoothing parameter).

At the time of the paper's publication, the use of external gazetteers was common in the task of NER. While it is an effective approach both in terms of computational cost (lookup table structure) and model performance (provides domain-specific knowledge), relying on such a resource is a symptom of the model failing to generalize effectively to the overall task of NER, regardless of the domain or language. The proposed neural architectures achieved state-of-the-art performance without relying on hand-crafted features and domain-specific knowledge. The LSTM-CRF model performed better than the chunking model, likely due to the context provided by the bidirectional LSTMs. However, it is important to note that the pretrained word embeddings gave the largest improvement, increasing the F1 score by +7.31 for the LSTM-CRF. Such pretrained embeddings are an external, language specific resource. Thus, while the authors made a significant leap in developing generalizable models for NER, there is still much work to be done in developing models that generalize effectively without relying on external, hand-crafted resources.

References

- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural Architectures for Named Entity Recognition. In *Proceedings of NAACL-HLT* (pp. 260-270). Association for Computational Linguistics.