Recall-Oriented Learning of Named Entities in Arabic Wikipedia (Supplemental Materials)

Behrang Mohit* **Nathan Schneider†** **Rishav Bhowmick*** **Kemal Oflazer*** **Noah A. Smith†**
School of Computer Science, Carnegie Mellon University
*P.O. Box 24866, Doha, Qatar †Pittsburgh, PA 15213, USA
{behrang@,nschneid@cs.,rishavb@qatar.,ko@cs.,nasmith@cs.}cmu.edu

Abstract

This document elaborates on the data, features, and modeling presented in (Mohit et al., 2012).

A Data

Table 1 summarizes the various corpora used in this work.

B Features

Our features include many that have been found to work well for Arabic in past research and some new features enabled by Wikipedia. We do not make use of any gazetteer, viewing the construction of a broad-domain gazetteer as a significant undertaking on its own and orthogonal to the challenges of a new text domain like Wikipedia.

We use a first-order structured perceptron; none of our features consider more than a pair of consecutive BIO labels at a time. The structured model enforces the constraint that NE sequences must begin with B (so the bigram $\langle O, I\rangle$ is disallowed).

Standard features. We include 15 contextual and lexical features capturing local context and shallow morphology. These consider a window of two previous words. They capture recurring parts of names, such as titles (e.g., *Dr.*) and descriptive nouns (e.g., *City* in *City of Cairo*). We also use the current word's length as some of the NEs have longer than average length. Following Abdul-Hamid and Darwish (2010), we extract a set of 12 character n-gram features: leading and trailing unigrams, bigrams, and trigrams starting from the first and the last two letters of the word (e.g., $\langle w_1, w_2\rangle$, $\langle w_2, w_3\rangle$, $\langle w_{n-2}, w_{n-1}\rangle$, and $\langle w_{n-1}, w_n\rangle$ are the bigrams extracted from a word with character sequence $\langle w_1, w_2, \ldots, w_n\rangle$). The character-level features capture prefixes and suffixes that typify names, particularly names transliterated into Arabic from other languages (such as *-man* in German surnames). Interior n-grams can match similar affixes occurring inside clitics such as the conjunction *wa-*, the definite article *Al-*, and the plural suffix *-At* (as our tokenizer does not separate these).

Morphology and shallow syntax. Arabic words generally carry rich morphological information, some of which (including noun-adjective agreement and special markings for construct-state nominals in compounds) is local to noun phrases such as NEs. For this reason, morphological features have been found to be useful in NE detection. Following Benajiba et al. (2008), we use the MADA toolkit (Habash and Rambow, 2005; Roth et al., 2008) to extract features encoding the normalized spelling, part-of-speech, aspect, case, number, gender, person, and definiteness/state of a word and its predecessor. As MADA was trained to achieve high performance on news text from the Arabic Treebank (Maamouri et al., 2004), we expect it to achieve somewhat lower accuracy on Wikipedia text. A small-scale evaluation of MADA's performance for Wikipedia data was encouraging, however; with regard to segmentation and diacriticization, less than 10% of MADA's output was judged faulty by both of our annotators.

Diacritics. Though most words in Arabic text are unvocalized, diacritics are commonly used to disambiguate names the first time they appear in an article. This feature indicates whether the current word had diacritics in the source text. (Diacritics are removed from the token for all other purposes.)

Projected English capitalization. As has been noted previously (Benajiba et al., 2008), capitalization is an extremely useful cue for NER in English; Arabic is at a disadvantage in this regard because the script does not specially mark proper names. To correct this we turn to lexi-

		documents	words	sents.	entities	MIS rate
Training	ACE+ANER	—	212,839	7,053	15,796	7%
	Wikipedia (unlabeled)	397	1,110,546	40,001	—	—
Development	ACE	—	7,776	250	638	3%
	Wikipedia (4 domains)	8	21,203	711	2,073	53%
Test	ACE	—	7,789	266	621	2%
	Wikipedia (4 domains)	20	52,650	1,976	3,781	37%
	history	5	13,046	381	1,158	6%
	science	5	15,151	667	882	61%
	sports	5	11,240	376	932	12%
	technology	5	13,213	552	809	83%

Table 1: Train and test corpora statistics for the ACE, ANER and Wikipedia articles. The last column records the proportion of annotated entity mentions belonging to the miscellaneous category.

cal correspondences between Arabic and English. One feature in our model indicates whether the word corresponds to a capitalized English word in MADA's Arabic-English lexicon. To improve recall in the Wikipedia domain, we construct a mapping between English and Arabic Wikipedia titles connected by cross-lingual links in article metadata.[1] Three indicator features encode whether the current word, the previous word, or the combination of the two map to a capitalized English term in this Wikipedia-derived lexicon.

C Models

Our starting point for statistical NER is a feature-based linear model over sequences, trained using the structured perceptron (Collins, 2002). This framework enables us to manipulate two key elements of the model: the features and the loss function used in training. It is closely related to the preponderance of recent research on statistical NER.

[1]From full snapshots of the Arabic and English versions of Wikipedia, we collect titles of 85,642 Arabic articles doubly cross-linked to an English article (i.e., the Arabic article has a cross-lingual link to the English article, and vice versa). As the first letter of the title itself is automatically capitalized, we decide whether to capitalize the English side of the entry based on a heuristic measure of how consistently the title term is capitalized within the article. (Specifically, we threshold on the ratio of capitalized to uncapitalized occurrences of the title term within the English article.) We chose this method for its simplicity; others have developed more sophisticated techniques, namely gazetteers constructed in a semi-automated fashion from Wikipedia and other resources (Benajiba et al., 2008; Shaalan and Raza, 2008; Attia et al., 2010).

C.1 Linear Feature-Based Model

Let $\boldsymbol{x} = \langle x_1, \ldots, x_M \rangle$ denote an input sequence, here a sentence in Arabic, and $\boldsymbol{y} = \langle y_1, \ldots, y_M \rangle$ denote a sequence of tags that encode named entity boundaries and labels ("BIO" tags, denoting tokens that are at the beginning, inside or outside of a NE). Given input $\boldsymbol{x}$, a linear model chooses an output

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \quad (1)$$

where $\mathbf{g}$ maps the input-output pair into $\mathbb{R}^D$, a D-dimensional feature space, and $\mathbf{w}$ is a weight vector in $\mathbb{R}^D$ that parameterizes the model. Equation 1 is known as the *decoding* problem.

In most NLP research with sequence models, $\mathbf{g}$ is designed to *factor* into local parts:

$$\mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{M+1} \mathbf{f}(\boldsymbol{x}, y_{i-1}, y_i) \quad (2)$$

This enables the use of a familiar dynamic programming technique—the Viterbi algorithm—for exactly solving the decoding problem in equation 1. The restriction is that each feature may only depend on two adjacent word-labels at a time. Such a model makes similar independence assumptions to those of a hidden Markov model.

C.2 Learning and the Perceptron

The perceptron can be understood in two ways: (i) as a simple iterative algorithm for finding a hyperplane (in D-dimensional feature space) that separates correct $\boldsymbol{y}$ values from incorrect ones, shown as Algorithm 1,[2] or (ii) as an empirical risk minimizer. Though the first view is more common, we

[2]The perceptron is guaranteed to eventually find such a hyperplane if one exists (Collins, 2002).

Input: data $\langle\langle \boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\rangle\rangle_{n=1}^{N}$; number of iterations T; rate schedule $\langle\alpha^{(t)}\rangle_{t=1}^{T}$
Output: $\mathbf{w}$
$\mathbf{w} \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  choose $\langle \boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\rangle$ u.a.r. from the data
  $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}^{(t)}, \boldsymbol{y})$ (eq. 1)
  if $\hat{\boldsymbol{y}} \neq \boldsymbol{y}^{(t)}$ **then**
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{(t)}(\mathbf{g}(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}) - \mathbf{g}(\boldsymbol{x}^{(t)}, \hat{\boldsymbol{y}}))$

Algorithm 1: Training with the perceptron.

adopt the second to help elucidate our new learning algorithm.

Given training examples $\langle\langle \boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}\rangle, \ldots, \langle \boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)}\rangle\rangle$, empirical risk minimization seeks:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} \ell(\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}, \mathbf{w}) \qquad (3)$$

where ℓ is some loss function, usually convex in $\mathbf{w}$, that penalizes mistakes.[3] The perceptron's familiar online updates (innermost line of Algorithm 1) can be understood as stochastic subgradient ascent on equation 3, with the perceptron's loss function:

$$\ell_{perceptron}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) = \max_{\boldsymbol{y}'} \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') - \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \qquad (4)$$

Alternative learning approaches use different loss functions (e.g., CRFs use the log loss). Structured SVMs (Tsochantaridis et al., 2004), notably, incorporate the notion of *cost* or error into the loss function. Let $c(\boldsymbol{y}, \boldsymbol{y}')$ denote a measure of error when $\boldsymbol{y}$ is the correct answer but $\boldsymbol{y}'$ is predicted. The structured hinge loss is $\ell_{hinge}(\boldsymbol{x}, \boldsymbol{y}, \mathbf{w}) =$

$$\max_{\boldsymbol{y}'} \left(\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + c(\boldsymbol{y}, \boldsymbol{y}')\right) - \mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}) \qquad (5)$$

The maximization problem inside the parentheses is known as *cost-augmented decoding*. If c factors similarly to equation 2, then we can increase penalties for $\boldsymbol{y}$ that have more local mistakes. This raises the learner's awareness about how it will be evaluated. Incorporating cost-augmented decoding into the perceptron is not a new idea (Gimpel and Smith, 2010a), and it relates closely to well-known "aggressive" algorithms for online max-margin learning (Crammer et al., 2006).

In NER, this extension can allow biasing the learner away from false negatives or false positives, in an unbalanced way (Gimpel and Smith, 2010b). Gimpel and Smith define word-local cost functions that differently penalize precision errors (i.e., $y_i = O \wedge \hat{y}_i \neq O$ for the ith word), recall errors ($y_i \neq O \wedge \hat{y}_i = O$), and entity class/position errors (other cases where $y_i \neq \hat{y}_i$). As we show (Mohit et al., 2012), a key problem in cross-domain NER is poor *recall*, so we will penalize recall errors most severely:

$$c(\boldsymbol{y}, \boldsymbol{y}') = \sum_{i=1}^{M} \begin{cases} 0 & \text{if } y_i = y'_i \\ \beta & \text{if } y_i \neq O \wedge y'_i = O \\ 1 & \text{otherwise} \end{cases} \qquad (6)$$

for a penalty parameter $\beta > 1$. We call our learner the "recall-oriented" perceptron (ROP), though more precisely it is stochastic subgradient ascent with the hinge loss (eq. 5) and the cost function in eq. 6. The change to Algorithm 1 is simply to use

$$\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}'} \left(\mathbf{w}^\top \mathbf{g}(\boldsymbol{x}, \boldsymbol{y}') + c(\boldsymbol{y}, \boldsymbol{y}')\right) \qquad (7)$$

which can be accomplished with a slight change to the Viterbi algorithm.

[3] In most machine learning approaches, a regularization term is added to avoid overfitting, e.g., $\|\mathbf{w}\|_2^2$. The perceptron does not explicitly regularize, relying instead on the stochasticity in the online updates and averaging or voting at the end of learning to avoid overfitting. We use averaging for our experiments.

References

Ahmed Abdul-Hamid and Kareem Darwish. 2010. Simplified feature set for Arabic named entity recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115, Uppsala, Sweden, July. Association for Computational Linguistics.

Mohammed Attia, Antonio Toral, Lamia Tounsi, Monica Monachini, and Josef van Genabith. 2010. An automatically built named entity lexicon for Arabic. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Yassine Benajiba, Mona Diab, and Paolo Rosso. 2008. Arabic named entity recognition using optimized feature sets. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 284–293, Honolulu, Hawaii, October. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings*

of the ACL-02 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, December.

Kevin Gimpel and Noah A. Smith. 2010a. Softmax-margin CRFs: Training log-linear models with loss functions. In *Proceedings of the Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736, Los Angeles, California, USA, June.

Kevin Gimpel and Noah A. Smith. 2010b. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University. `http://www.lti.cs.cmu.edu/research/reports/2010/cmulti10008.pdf`.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109.

Behrang Mohit, Nathan Schneider, Rishav Bhowmick, Kemal Oflazer, and Noah A. Smith. 2012. Recall-oriented learning of named entities in Arabic Wikipedia. In *Proceedings of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, Avignon, France, April. Association for Computational Linguistics.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of ACL-08: HLT*, pages 117–120, Columbus, Ohio, June. Association for Computational Linguistics.

Khaled Shaalan and Hafsa Raza. 2008. Arabic named entity recognition from diverse text types. In *Advances in Natural Language Processing*, pages 440–451. Springer.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In Russ Greiner and Dale Schuurmans, editors, *Proceedings of the 21st International Machine Learning Conference (ICML)*, Banff, Canada. ACM.