LingPipe Blog

Natural Language Processing and Text Analytics

Coding Chunkers as Taggers: IO, BIO, BMEWO, and BMEWO+

By Breckbaldwin

I've finished up the first order linear-chain CRF tagger implementation and a bunch of associated generalizations in the tagging interface. Now it's time to code up chunkers using CRFs, and I'm faced with the usual problem of how to encode chunks, which are about character spans, as taggings, which are sequences of tokens (words or other units) and tags (aka labels, categories).

An Example

Here's an example of a tokenized string and its encoding using several standards:

Tokens	**IO**	**BIO**	**BMEWO**	**BMEWO+**
Yesterday	O	O	O	BOS_O
afternoon	0	O	O	O
,	0	O	O	O_PER
John	I_PER	B_PER	B_PER	B_PER
J	I_PER	I_PER	M_PER	M_PER
.	I_PER	I_PER	M_PER	M_PER
Smith	I_PER	I_PER	E_PER	E_PER
traveled	O	0	O	PER_O
to	O	O	O	O_LOC
Washington	I_LOC	B_LOC	W_LOC	W_LOC
.	O	O	O	O_EOS

IO Encoding

The simplest encoding is the IO encoding, which tags each token as either being in (I_*X*) a particular type of named entity type *X* or in no entity (O). This encoding is defective in that it can't represent two entities next to each other, because there's no boundary tag.

BIO Encoding

The "industry standard" encoding is the BIO encoding (anyone know who invented this encoding?). It subdivides the in tags as either being begin-of-entity (B_*X*) or continuation-of-entity (I_*X*).

BMEWO Encoding

The BMEWO encoding further distinguishes end-of-entity (E_*X*) tokens from mid-entity tokens (M_*X*), and adds a whole new tag for single-token entities (W_*X*). I believe the BMEWO encoding was introduced in Andrew Borthwick's NYU thesis (http://citeseer.ist.psu.edu/old/699859.html) and related papers on "max entropy" named entity recognition around 1998, following Satoshi Sekine's similar encoding for decision tree named entity recognition. (Satoshi and David Nadeau just released their Survey of NER (https://lingpipe-blog.com/2009/07/14/nadeau-and-sekine-2007-a-survey-of-named-entity-recognition-and-classification/).)

BMEWO+ Encoding

I introduced the BMEWO+ encoding for the LingPipe HMM-based chunkers (http://alias-i.com/lingpipe/docs/api/com/aliasi/chunk/HmmChunker.html). Because of the conditional independence assumptions in HMMs, they can't use information about preceding or following words. Adding finer-grained information to the tags themselves implicitly encodes a kind of longer-distance information. This allows a different model to generate words after person entities (e.g. John *said*), for example, than generates words before location entities (e.g. *in* Boston). The tag transition constraints (B_*X* must be followed by M_*X* or E_*X*, etc.) propagate decisions, allowing a strong location-preceding word to trigger a location.

Note that it also adds a begin and end of sequence subcategorization to the out tags. This helped reduce the confusion between English sentence capitalization and proper name capitalization.

Transition and Feature Tying

Consider the BMEWO notation. At least for regular named-entity recognition in text, we probably don't want to distinguish being preceded by an end-of-entity token labeled E_*X* from a whole entity token labeled W_*X*.

In HMMs, we'd tie transitions, so p(*T*|E_*X*) = p(*T*|W_*X*) for all tags *T*.

In CRFs, we could tie the feature generation process, so having either preceding tag (end or whole entity) produces the same features, and hence the same probability estimates.

Complexity

First-best decoding in a first-order CRF or HMM is quadratic in the number of tags (and linear in the number of tokens). Here's a rundown:

Encoding	**# Tags**	**N=1**	**N=3**	**N=20**
IO	N+1	2	4	21
BIO	2N+1	3	7	41
BMEWO	4N+1	5	13	81
BMEWO+	7N+3	10	24	143

Legal Transition Pruning

It's not quite as bad as number of tags squared in practice, because not every tag can follow every other tag in the more complex codings. Even in BIO encoding, I_*X* can only follow B_*X*. In BMEWO encoding, M_*X* and E_*X* can only follow B_*X* or I_*X*, and O can only follow O, E_*X*, or W_*X*.

Of course, this is a major pain to compute and specify in the API, as you need some way of figuring out legal transitions. For HMMs, I used the fact that transition probabilities were zero to rule out consideration. I'll have to add some way to specify legal transitions if I implement something general for CRF tagger decoding.

Necessary for CRFs?

Jenny Finkel (http://www.stanford.edu/~jrfinkel/) recently told me she really thought IO encoding is all you need in most cases. It's how I coded a single-entity problem for the mechanical Turkers for NE annotation. As far as encodability, it's fine for MUC, almost OK for Biocreative [there are a couple contiguous entities, I think], but degenerate in general. The really nice thing about this encoding is that for a single entity type, we can get away with a single feature vector.

The recent Ratinov and Roth NE survey (https://lingpipe-blog.com/2009/07/09/ratinov-and-roth-2009-design-challenges-and-misconceptions-in-named-entity-recognition/) concluded BMEWO was superior to BIO for a second-order online logistic model (basically what's known as a MEMM, but with best guess at each token enforced going forward).

I found BMEWO+ worked best for single-entity and small entity set tagging for our HMMs.

What to Do for First-Order CRFs?

What I don't know is what I should do for CRFs. I've settled on only using first-order CRFs, and not having feature extraction depend on the output tag (or more precisely, every feature is crossed with every output tag at every position).

I'm tempted to try to punt and write up a flexible TaggingChunkingCodec interface and let the user specify it. The real problem is that evaluation's not simple in this context, because results vary pretty dramatically based on features. That's why I love and hate CRFs ().

This entry was posted on October 14, 2009 at 2:47 pm and is filed under Carp's Blog, LingPipe in Use, LingPipe News. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

2 Responses to "Coding Chunkers as Taggers: IO, BIO, BMEWO, and BMEWO+"

Martin Says:
February 6, 2010 at 3:30 am | Reply
Re: Who invented the BIO coding?

As far as I can tell, the BIO coding scheme goes back to Adwait Ratnaparkhi's 1998 PhD thesis (p. 57ff.). There the labels are known as Start, Join, and Other. The connection between Ratnaparkhi's thesis and BIO/IOB2 scheme is made explicit in a 1999 EACL paper by Tjong Kim Sang et al. ("Representing Text Chunks"). That paper discusses several other encoding schemes as well.

lingpipe Says:
February 6, 2010 at 7:16 pm | Reply
I just took a look around and found at least one earlier reference, Ramshaw and Marcus's 1995 paper Text chunking using transformation-based learning (section 4.1, "Encoding Choices"). They don't cite any earlier sources.