# LingPipe for 99.99% Recall of Gene Mentions

**Bob Carpenter**
carp@alias-i.com

Alias-i, Inc., 181 North 11th St., Brooklyn, NY 11211, USA

**Abstract**

Text data mining over biomedical research literature is a needle-in-a-haystack problem. We contend that first-best methods performing at 90% F-measure are insufficient, especially given that performance is much worse for "unseen" phrases. In this paper, we recast the problem as one of $n$-best search rather than first-best database population. We describe LingPipe's HMM and character language model-based chunkers, which extract mentions of genes in unseen MEDLINE abstracts at 99.99% recall with greater than 50% mean-average precision. We provide evaluation results in terms of received precision-recall curves on unseen data.

**Keywords:** named entity extraction, confidence ranking, text data mining, search, character language models, hidden Markov models, forward-backward algorithm, A* algorithm

## 1 Introduction

Using a first-best entity extractor is akin to removing Google's "Search" button and relying on "I'm Feeling Lucky". Even with state-of-the-art precision, recall is going to be unacceptable for individual research or data mining purposes, which are often of the needle-in-a-haystack variety. Researchers don't need to find dozens or hundreds of references to a common pathway interaction, they need to find the rare references that link two of the genes that are differentially expressed in a series of microarray assays in an unexpected way.

Evaluations by F-measure overemphasize performance on common, oft-repeated mentions. When performance is reported on mentions not included in the training data, error rates typically double or more. The alternative we offer is $n$-best output with conditional probability estimates of the mention given the text. This normalizes scores across sentences and documents, allowing the annotation problem to be recast as a search problem. We believe that scoring metrics for search, such as mean average precision and area under the receiver operating characteristic curve, are more appropriate for evaluating real-world uses of text data mining.

LingPipe's confidence-based chunkers are first-order hidden Markov models with emission probabilities estimated by (padded) character language models. Using a generalized form of best-first search over the lattice produced by the forward-backward algorithm, these chunkers are able to iterate an arbitrary number of chunks in confidence-ranked order. Setting the threshold to 99.999% recall, these chunkers run at 330,000 characters/second.

LingPipe also contains a longer-distance character-language-model based chunker that rescores $n$-best whole-sentence analyses from the confidence-based chunker. We submitted a run of that chunker to BioCreAtIvE, as well as confidence-based results. See [2] for a description of the rescoring model.

## 2 LingPipe's Character Language Models

LingPipe's classification, tagging, and entity extraction are all based on $n$-gram character language models. Language models define probability distributions $p(\sigma)$ over strings $\sigma \in \Sigma^*$ drawn from a fixed alphabet of characters $\Sigma$. LingPipe adopts a standard random process approach to $n$-gram language models, where probabilities are normalized over strings of a fixed length.

The process models factor the probability $p(\sigma c)$ of the string $\sigma$ followed by the character $c$ using the chain rule: $p(\sigma c) = p(\sigma) \cdot p(c|\sigma)$. The $n$-gram Markov assumption restricts the context of a conditional estimate $p(c|\sigma)$ to the last $n-1$ characters of $\sigma$, taking $p(c_n|\sigma c_1 \cdots c_{n-1}) = p(c_n|c_1 \cdots c_{n-1})$.

The maximum likelihood estimator for this model is $\hat{p}_{ml}(c|\sigma) = \mathsf{count}(\sigma c)/\mathsf{extCount}(\sigma)$, where $\mathsf{count}(\sigma)$ is the raw corpus count of the string $\sigma$ and $\mathsf{extCount}(\sigma) = \sum_c \mathsf{count}(\sigma c)$ is the number of single character extensions of $\sigma$.

LingPipe interpolates all orders of maximum likelihood estimates using Witten-Bell smoothing. The smoothed estimates are defined by $\hat{p}(c|d\sigma) = \lambda(d\sigma)\hat{p}_{ml}(c|d\sigma) + (1 - \lambda(d\sigma)\hat{(}p)(c|\sigma)$ with the boundary condition $\hat{p}() = 1/\mathsf{size}(\Sigma)$ given by the uniform distribution. Witten and Bell smoothing takes the interpolation ratio $\lambda(\sigma) = \mathsf{extCount}(\sigma)/(\mathsf{extCount}(\sigma) + \theta \cdot \mathsf{numExts}(\sigma))$, where $\mathsf{numExts}(\sigma) = \mathsf{size}(\{c|\mathsf{count}(\sigma c) > 0\})$. The free parameter $\theta$, which controls the degree of smoothing, was fixed at 1.0 by Witten and Bell, but is set to the $n$-gram order by default in LingPipe.

Bounded language models assume distinct begin-of-string (`BOS`) and end-of-string (`EOS`) string markers, setting $\hat{p}(\sigma) = \hat{p}(\sigma$ `EOS`$|$`BOS`$)$, where the conditional probaility is estimated using a process model. With string boundary padding, normalization is over all strings, with $\sum_{\sigma \in \Sigma^*} \hat{p}(\sigma) = 1$.

## 3 HMMs with Character Language Model Emissions

LingPipe employs first-order HMMs for tagging, where the hidden states, as usual, correspond to tags. Taggers assume a tokenization scheme that deterministically breaks an input into sequences of tokens. The joint probablity of a token sequence $\sigma_1, \ldots, \sigma_n$ and tag sequence $t_1, \ldots, t_n$ is defined by $p(\sigma_1, \ldots, \sigma_n, t_1, \ldots, t_n) = p(t_1, \ldots, t_n) \cdot p(\sigma_1, \ldots, \sigma_n | t_1, \ldots, t_n)$. A first-order HMM defines $p(t_1, \ldots, t_n) = p_{start}(t_1) \cdot \prod_{i>1} p(t_i|t_{i-1}) \cdot p_{end}(t_n)$; note the special estimates for start and end tags, which ensures the sum of all token/tag sequences is 1.

In typical HMMs, emissions are estimated as multinomials, with some kind of special handling for unseen tokens. LingPipe's HMMs are unusual in that they estimate the probability $p(\sigma|t)$ of the token $\sigma$ given the tag $t$ using bounded character language models, one for each tag $t$. This has the advantage of including general $n$-gram subword features within a fully generative probability model, as well as defining a proper probability model normalized over the infinite set of possible string emissions.

LingPipe's HMMs come with three decoders. The first is a standard Viterbi first-best decoder. The second is a standard $n$-best decoder, which applies a Viterbi pass in a forward stage and then uses these as A* estimates to perform an exact backward search to iterate over an arbitrary number of unnormalized estimates of $p(t_1, \ldots, t_n | \sigma_1, \ldots, \sigma_n)$. The third decoder is a forward-backward decoder, which computes conditional probabilities of a tag given an input sequence.

Consider input tokens $\sigma_1, \ldots, \sigma_n$. The forward value for a tag $t$ and input position $i$ is $\mathsf{fwd}(t, i) = p(\sigma_1, \ldots, \sigma_{i-1}, \mathsf{tag}(i) = t)$, which is the probability of the first $i-1$ input tokens resulting in the token $\sigma_i$ at position $i$ being assigned tag $t$. This value is estimated in linear time using the forward algorithm, at each stage computing the forward value as the sum of the values of all transitions from the previous forward values. Backward values for position $i$ and tag $t$ are defined by $\mathsf{bk}(t, i) = p(\sigma_i, \ldots, \sigma_n | \mathsf{tag}(i) = t)$, the conditional probability of the current and remaining tokens given that the current tag is $t$. Backward probabilities are also easily computed in a single linear-time pass. Multiplying the forward and backward values produces the joint probability of a tag given an input sequence, $p(\sigma_1, \ldots, \sigma_n, \mathsf{tag}(i) = t) = \mathsf{fwd}(t, i) \cdot \mathsf{bk}(t, i)$. The conditional probabiliy of position $i$ receiving tag $t$ is derived by marginalization, $p(\mathsf{tag}(i) = t | \sigma_1, \ldots, \sigma_n) = p(\sigma_1, \ldots, \sigma_n, \mathsf{tag}(i) = t) / \sum_{t'} p(\sigma_1, \ldots, \sigma_n, \mathsf{tag}(i) = t)$.

## 4 HMM Encodings for Chunking with Confidence

It is common to encode a chunking problem, such as named entity extraction, as a tagging problem. The typical tag set for a task like BioCreAtIvE would involve three tags: $B_G$ for the first token in a gene mention, $I_G$ for otehr tokens in a gene mention, and $O$ for tags that are not part of a gene mention. It is possible to assign chunk probabilities with these tags, but the algorithm is tricky because of the lack of end markers [3]. This encoding is also problematic for our first-order HMMs; they tend to have difficulty finding boundaries, especially end boundaries.

We solve the search and estimation together using an encoding that is sensitive to position, using tags $B_G$ (first token in mention), $M_G$ (internal token in mention), $E_G$ (last token in mention), and $W_G$ (single token mention). Furthermore, we subcategorize the non-gene tags the same way ($B_O$, $M_O$, $E_O$ and $W_O$). This distinguishes the first and last words in gene mentions, as well as the words directly preceding and following a gene mention.

With this encoding, the conditional probability of a subsequence of tokens being a gene mention given the entire sequence, $p(\sigma_i, \ldots, \sigma_k : G | \sigma_1, \ldots, \sigma_n)$, is:

$$\mathsf{fwd}(B_G, i) \cdot \hat{p}(\sigma_i | B_G) \cdot ( \prod_{i<j<k} \hat{p}(\sigma_j | M_G) \cdot \hat{p}(M_G | M_G) \ ) \cdot \hat{p}(E_G | M_G) \cdot \mathsf{bk}(E_G, k)$$

The probability of a single token gene mention is just the conditional tag probability, which is the product of the forward and backward estimates. LingPipe iterates the chunks in conditional probability order using an exact best-first search that keeps all partial entities on a priority queue, always expanding the one with highest probability, and popping and returning an answer when one is found.

## 5 Results on BioCreAtIvE II Gene Mention Data

LingPipe was trained on the BioCreAtIvE II data (see [4] and this volume), using default settings. Given the sentence *p53 regulates human insulin-like growth factor II gene expression through active P4 promoter in rhabdomyosarcoma cells*, the phrases extracted as chunks and their conditional probability estimates are *p53*:.999, *P4 promoter*:.733, *insulin-like growth factor II gene*:.606, *human insulin-like growth factor II gene*:.382, *active P4 promoter*:.140, *P4*:.092, *active P4*:.009, *insulin-like growth factor II*:.007, *human insulin-like growth factor II*:.004.

The precision versus recall curve is as follows:

| Recall | .02 | .10 | .20 | .30 | .40 | .50 | .60 | .70 | .80 | .90 | .95 | .99 | .999 | .9999 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Precision | .83 | .76 | .72 | .69 | .65 | .61 | .54 | .46 | .36 | .25 | .18 | .11 | .08 | .07 |

This curve is computed by sorting all genes output in confidence order and then moving down the list, computing precision and recall at each point; average precision is the average of the values at true positive positions. For instance, LingPipe extracts 95% of all gene mentions in a list with 18% precision, and 99.99% of all mentions with 7% precision.

## References

[1] Alias-i. 2006. LingPipe 2.3.0. `http://www.alias-i.com/lingpipe`. (BioCreAtIvE II in sandbox).

[2] Carpenter, B. 2006. Character language models for Chinese word segmentation and named entity recogntion. *Proceedings of the 5th ACL Chinese Special Interest Group.*

[3] Culotta, A. and A. McCallum. 2004. Confidence estimation for information extraction. *NAACL.*

[4] Tanabe, L, N. Xie, L. H. Thom, W. Matten, and W. J. Wilbur. 2005 *BMC Bioinformatics* **6**(Suppl 1):S3.