# Bayesian Text Segmentation for Index Term Identification and Keyphrase Extraction

David Newman,[♢] Nagendra Koilada,[♢] Jey Han Lau[♡♠] and Timothy Baldwin[♡♠]
♢ Dept of Computer Science, University of California, Irvine, USA
♡ Dept of Computing and Information Systems, The University of Melbourne, Australia
♠ NICTA Victoria Research Laboratory, Australia
newman@uci.edu, nkoilada@uci.edu, jhlau@csse.unimelb.edu.au, tb@ldwin.net

ABSTRACT
Automatically extracting terminology and index terms from scientific literature is useful for a variety of digital library, indexing and search applications. This task is non-trivial, complicated by domain-specific terminology and a steady introduction of new terminology. Correctly identifying nested terminology further adds to the challenge. We present a Dirichlet Process (DP) model of word segmentation where multiword segments are either retrieved from a cache or newly generated. We show how this DP-Segmentation model can be used to successfully extract nested terminology, outperforming previous methods for solving this problem.

KEYWORDS: Dirichlet Process, segmentation, Bayesian learning, index term, keyphrase extraction.

## 1 Introduction

Index terms and keyphrases are widely used to enhance the browsing/accessibility of documents, especially scientific literature (Gutwin et al., 1999). Index terms are the main terms and concepts in a document, as would be contained in the index of a book or edited collection of papers, for example. They typically number in the hundreds for a single volume, and are conventionally annotated for their key occurrences in the document. Keyphrases are assigned to each document in the form of a handful of terms which capture the key topics or concepts covered in the document. Keyphrases are commonly used as a means of clustering/linking documents in digital libraries. While there is no question of the utility of both index terms and keyphrases, annotating a document with index terms is an arduous and largely unrewarding task, and annotators are notoriously inconsistent at index term and keyphrase assignment (Leonard, 1977; Kim et al., 2010). As such, there is a real need for computational methods which both speed up and homogenize the assignment of index terms and keyphrases.

The focus of this paper is an unsupervised methodology for automatically identifying both index terms in a document collection and keyphrases for a single document. The methodology is called Dirichlet Process Segmentation (DP-seg) and was originally developed by Goldwater et al. (2009) for the task of word segmentation over phoneme streams, in the context of child language acquisition. DP-seg is particularly well suited to the tasks of index term identification and keyphrase extraction because: (1) it is able to identify individual type-level occurrences of a given term; (2) it is context-sensitive and able to robustly deal with "nested" multiword terminology (e.g. are *support vector* and *vector machine* also index terms, or just *support vector machine*?); (3) it is highly sensitive and can be applied to small-scale corpora (important for keyphrase extraction, where the input may be a single document); and (4) it is highly parameterizable.

Our contributions in this paper are: (1) the novel application of DP-seg to the tasks of index term identification and keyphrase extraction; (2) construction of a number of index term identification datasets spanning a number of scientific domains; and (3) extensive empirical comparison with standard methods for multiword extraction, and demonstration of the empirical superiority of DP-seg.

## 2 Related Work

There are several closely related language modeling tasks in this area that differ in important ways. Here we further clarify what tasks we are addressing in this paper, and review related work for each.

We define a multiword term (i.e. $n$-gram, where $n \geq 2$) as any contiguous sequence of terms in a sentence from some text. We use the usual convention of *types* being the list of distinct multiword terms (at a document or document collection level), and *tokens* being the specific appearances of those multiword terms in the text. Note that we can extract *types* and/or *tokens* either from a single document, or a collection of documents, and our focus for this work will be collection/corpus-based extraction.

Collocation extraction is a well known task that has been extensively researched. In collocation extraction, the goal is to extract a list of types from a single document, often of a particular type (e.g. as defined by a POS tag sequence: Evert (2004); Pecina (2008)). Evaluation is done by comparing to a gold standard produced by lexicographers. Identification of multiword expressions (MWEs) is a related task, but operates at the token level, often relative to a predetermined MWE type or lexicon of MWE candidates (Kim and Baldwin, 2010; Baldwin and Kim, 2009). Arguably the closest task to what we are doing is Keyphrase extraction, and indeed our content of interest is scientific literature (Witten et al., 1999; Kim et al., 2010). Keyphrase extraction operates over individual documents, often based on corpus-level lexical analysis. In this paper, in addition to experimenting over keyphrase extraction as conventionally defined, we introduce the additional task of index term extraction.

Extraction of index terms from a corpus is important and useful for a wide variety of digital library and scholarly applications. It extends keyphrase extraction (which is typically performed on a single article or document) to the broader setting of extracting all concepts from a book or collection of papers. Thus index terms tend to be more numerous than keyphrases (the index of a textbook may list hundreds of index terms). Unlike keyphrases, index terms cover all concepts appearing in text — from fairly general concepts (e.g. *training data*) to specific concepts (e.g. *textual entailment*). We are interested in both type-level extraction (for the index) and token-level identification (to be able to identify the individual occurrences in the text) of index terms. An important aspect of our task is uncovering the correct *nesting*. Extraction methods need to correctly determine when to choose a shorter and more general *n*-gram (e.g. *machine translation*) vs. a longer and more specific *n*-gram (e.g. *statistical machine translation*), while avoiding extracting nonsensical *n*-grams (e.g. *statistical machine*).

Keyphrase extraction is closely related to index term extraction. Recently, there was a shared task on keyphrase extraction in SemEval-2010 (Kim et al., 2010), where over twenty teams submitted systems to extract keyphrases from a set of 244 computer science articles. Evaluation was done against a matching set of gold standard keyphrases. We include this SemEval dataset in our experiments, and compare to those published results.

Perhaps the closest related work to our current work is the c/nc-value algorithm (Frantzi et al., 2000), which is designed for extracting and ranking a list of terminology types from a corpus. The c/nc-value algorithm specifically finds and ranks *nested* multiword terminology (such as *support vector* as well as *support vector machine*). The c/nc-value algorithm is a heuristic that takes as input the exhaustive list of 2- through 5-gram noun chunks. The c-value is a score that is an “effective” frequency, which includes an upweighting of $\log n$ for *n*-grams, to take into account the geometrically fewer *n*-grams as $n$ increases. The algorithm also includes a discounting of *n*-grams that appear in longer *n*-grams. The goal of the algorithm is to discover all nested terminology. The nc-value part of the algorithm is an extra boosting of the signal by using context information. Frantzi et al. showed c/nc-value to work well for producing the lexicon of index term types from a relatively small set of documents.

## 3 Bayesian Segmentation

Our DP-seg model is a direct adaptation of the Bayesian model of Goldwater et al. (2009). In that work, Goldwater et al. took a stream of phonemes to discover English words, with the goal of correctly identifying word boundaries. We adapt their model and apply it to the problem of taking a stream of words, and finding suitable boundaries between multiword segments. To provide a context of what is an appropriate multiword segment, we have defined our goal as identifying and extracting index terms (i.e. terms that would plausibly appear in an index). We note that our multiword version of this problem operates on a much larger scale than the multi-phoneme problem. Instead of learning a lexicon of 1000s of words from approximately 100 phonemes, we are learning a lexicon of possibly millions of multiword segments from a list of $\sim$100,000 words.

We present and describe the DP-seg model almost verbatim from Goldwater et al. (2009). The model assumes that the sequence of $n$-grams or multiword segments (MWE) in the corpus is generated using the following four steps:

1. Generate the number of MWE types that will be in the MWE lexicon
2. Generate the token frequency for each MWE type
3. Generate the word-sequence representation of each MWE type
4. Generate an ordering for the set of MWE tokens.

In the unigram version of the model where multiword segments are statistically independent, the i$^{th}$ MWE is generated as follows:

1. Decide if mwe$_i$ is a novel lexical item
2. **(a)** If so, generate an MWE form (words $w_1 \ldots w_M$) for mwe$_i$
   **(b)** If not, choose an existing lexical form $x$ for mwe$_i$

We assign probabilities to each of the above steps as follows:

1. P(mwe$_i$ is novel) $= \frac{\alpha}{n+\alpha}$, P(mwe$_i$ is not novel) $= \frac{n}{n+\alpha}$
2. **(a)** $P(\text{mwe}_i = w_1 \ldots w_M | \text{ mwe}_i \text{ is novel}) = p_\#(1-p_\#)^{M-1}\Pi_{j=1}^{M} P(w_j)$
   **(b)** $P(\text{mwe}_i = x | \text{ mwe}_i \text{ is not novel}) = \frac{n_x}{n}$

where $\alpha$ is a parameter of the model, $n$ is the number of previously generated MWEs, $n_x$ is the number of times lexical item $x$ has occurred in those MWEs, and $p_\#$ is the probability of generating a segment boundary. We combine these probabilities to get the following expression for the distribution over mwe$_i$ given mwe$_{-i}$, the previous MWE:

$$P(\text{mwe}_i = x|\text{mwe}_{-i}) = \frac{n_x}{i-1+\alpha} + \frac{\alpha P_0(\text{mwe}_i = x)}{i-1+\alpha} \tag{1}$$

where $P_0$ is the probability of generating the MWE anew, as per Step 2(a) above. Note that in all of the above, the MWE can consist of a single word.

As Goldwater et al. point out, this model is quite intuitive. In Step 1, when $n$ is small, we are more likely to generate a new multiword segment. As $n$ grows, we increasingly tend to retrieve the multiword segment from the cache. Also, the model discourages long segments through longer products of probabilities. By having the likelihood of retrieving a multiword from the cache depend on its frequency, we produce a power-law distribution of multiword types, which is appropriate for language modeling.

This model is a type of Dirichlet Process, a popular tool in Bayesian statistics for nonparametric modeling, where the dimensionality of our MWE lexicon grows with the size of the data. The Dirichlet Process has two parameters, the concentration parameter $\alpha$ and the base distribution $P_0$. Formally, $\text{mwe}_i|G \sim G$, $G|\alpha, P_0 \sim \text{DP}(\alpha, P_0)$. As is customary, we integrate out $G$, and estimate the conditional probability of sampling a MWE by the Chinese Restaurant Process (CRP). Simply stated, the CRP is where the i$^{th}$ customer entering a Chinese Restaurant sits at table $i$ with probability proportional to the number of people sitting at table $i$, or sits at a new table with probability proportional to $\alpha$. Note this generation of MWEs is the two stage adaptor grammar framework described by Goldwater et al. (2005), with $P_0$ as generator, and CRP as the adaptor. The CRP is the basis of the cache model: One either generate MWEs by either retrieving from cache (existing table), or generate anew (new table).

## 3.1 Inference

For inference, we use the Gibbs sampling procedure presented by Goldwater et al. The Gibbs sampler considers one possible segment boundary point at a time, so each sample is from a set of two hypotheses, $H_0$ and $H_1$. These hypotheses contain all the same segment boundaries except at the one position under consideration, where $H_1$ has a boundary and $H_0$ does not. To sample a hypothesis, we calculate the relative probabilities of $H_0$ and $H_1$. Since $H_0$ and $H_1$ are the same except for a few rules, this is straightforward. Let $H^-$ be all of the structure and data shared by the two hypotheses, including $n$ words. Then, given MWEs $w_1$ and $w_2$ and the combined MWE $w_{12}$, using Equation 1 we get:

$$P(H_0|H^-) = \frac{n(w_{12}) + \alpha P_0(w_{12})}{n + \alpha} \tag{2}$$

$$P(H_1|H^-) = \frac{n(w_1) + \alpha P_0(w_1)}{n + \alpha} \cdot \frac{n(w_2) + I(w_1 = w_2) + \alpha P_0(w_2)}{n + 1 + \alpha} \tag{3}$$

During preliminary experiments, we learned segmentations with DP-seg using between 5000 and 20000 sweeps through each corpus. We performed convergence checks to determine an appropriate number of iterations of the Gibbs sampler. Convergence can depend on thematic coherence and phrasing redundancy in the text (for example, a collection of abstracts from a PubMed search will have more repeated $n$-grams than a similar sized collection of full-text articles). By creating an approximate gold standard (using 20,000 iterations and averaging over 10 differently initialized runs), we determined that 5,000 iterations of the Gibbs sampler produced reasonably well-converged segmentations. For all experiments, except for those investigating the effect of parameters, we used Goldwater's settings of $\alpha = 20$ and $p_\# = 0.5$. We also used an annealing schedule with initial exponent of 0.1, going up to a exponent of 1 after 90% of the iterations, then no annealing for the final 10% of the iterations.

## 3.2 Pre-Processing

As with many NLP tasks, we have a variety of choices for preprocessing the raw input text. Some of these choices include whether to lemmatize (including stemming as a crude form of lemmatization), whether to POS tag, and whether to remove frequently occurring stopwords. Because we compare DP-seg to c/nc-value algorithm results, for the majority of experiments we selected a preprocessing strategy that was consistent with the input required by the c/nc-value algorithm. We used OpenNLP for lemmatization and POS tagging, and filtered out stop-words from a very limited list of common stopwords consisting of prepositions and determiners (fewer than 100 stopwords in total). Note that this would make terminology such as *part of speech* appear as *part speech*. Since DP-seg does not require *any* preprocessing at all, for some experiments, we omitted all preprocessing to examine its effect.[1]

## 4 Datasets

We used a combination of new and existing datasets for our experiments. Since there are no readily-available gold-standard datasets for index terms, we generated our own. What constitutes an index term can be subtle and debatable: some index terms are obvious, such as *semantic role labeling*; others are less so, e.g. *training data* is a concept, albeit general, that

[1]Note, however, that preprocessing has a significant impact on runtime for DP-seg: the larger the vocabulary and longer the texts, the longer the runtime.

| Collection | Source | Num. Docs | Num. Words |
|---|---|---|---|
| ACL | ACL Corpus | 17k fulltext | 40M |
| NIPS | NIPS Corpus | 2k fulltext | 5.4M |
| DNA | PubMed search results | 60k abstracts | 7.0M |
| HGT | PubMed search results | 7k abstracts | 0.7M |

Table 1: Description and sizes of the four document collections.

could reasonably be found in an index, while *data point* is probably too generic to be found in most indexes.

## 4.1 Document Collection Data

Our experiments start with four document collections from scientific/technical literature, described in Table 1. The ACL corpus consists of fulltext research articles from computational linguistics (Radev et al., 2009). The NIPS corpus consists of fulltext articles in machine learning and neurobiology (from `books.nips.cc`). The final two document collections are search results sets from PubMed (`www.pubmed.gov`). The DNA collection contains titles and abstracts in the search results from the query *dna microarray*. The HGT collection contains titles and abstracts in the search results from the query *horizontal gene transfer*. All four of these collections are rich in technical terminology and nested index terms, and thus ideal for experimentation.

### 4.1.1 Creating Gold Standard

We created a list of gold standard index terms from our four datasets to evaluate the performance of DP-seg. Because of the size of these datasets (over 80K documents and 50M words), it was not possible to annotate an exhaustive/comprehensive list of index terms. Therefore we used a pooling strategy to create a gold standard from top-ranked candidate terms, using the top-1000 index terms from DP-seg, c/nc-value and the Student's t-test (a baseline lexical association measure for collocation extraction), restricted to noun chunks. Note that the ranking for c/nc-value and t-test is based directly on the scores returned by the methods, while the ranking for DP-seg is based on the token-level frequency of each term in the output.

We used the union of all of these candidate index terms for annotation, and randomized the set so that annotators had no insights into which candidates came from which method. A preliminary observation of this list indicated that these top-ranked index terms were dominated by bigrams. Since we were interested in understanding the performance of DP-seg and c/nc-value on nested terminology, we also included for annotation the top-100 4-grams and 3-grams, and all 3-gram and 2-gram subphrases for each.

We used a pool of 25 annotators to annotate subsets of the combined list of 6500 candidate index terms, requesting a binary judgment on whether the candidate term was a plausible index term. We had a minimum of 3 annotators per candidate index term, and an average of 4.4 annotations per candidate index term. In cases of disagreement, we took the majority decision. Our instructions to annotators stated:

> For each of the following expressions, judge its appropriateness as an "index term" (as in it would be appropriate for inclusion in the index of a book or edited volume of papers). The expression has a

well-defined standalone meaning and would be appropriate for inclusion in an index of terms for that domain (e.g. *decision tree*, in the case of the machine learning domain, but not *observable markov* as it is an incorrectly-extracted sub-string of something like *partially observable markov decision process*).

The annotation tasked proved reasonably straightforward, with an inter-annotator agreement of 0.65 using Fleiss' kappa, suggesting substantial agreement.

### 4.2 SemEval Keyphrase Extraction Data

In addition to our four collections of index terms, we used an existing dataset for keyphrase extraction evaluation — the SemEval-2010 keyphrase extraction data. The SemEval data is a collection of 244 scientific articles released as part of a shared task for keyphrase extraction (Kim et al., 2010). In each document, there is a set of author- and reader-assigned keywords, and all keyphrases are stemmed using the English Porter stemmer. The gold standard keyphrases for each document are largely extracted from the document text, although approximately 15%-19% of the keyphrases do not appear in the paper.

To evaluate the participating systems, micro-averaged F-scores are calculated over the top-5/10/15 candidates. A candidate is considered a match if it is an exact match with one of the gold standard keyphrases; partial matching is not considered in the evaluation.

Note that there is a subtle difference between the two tasks: index term identification is annotated/evaluated at the *document collection* level (separately for each of our four document collections), while keyphrase extraction is annotated/evaluated at the *document* level. Another reason for evaluating our method over keyphrase extraction is that the task is well established, and we can benchmark DP-seg against results for pre-existing systems tuned to the task.

## 5 Results

Standard evaluation for this type of task is based on precision (P), recall (R) and F-score (F), at the level of either types or tokens. In our index term data, we don't have exhaustive annotations, meaning we aren't able to evaluate recall. Instead, we use precision at $N$ (P@$N$), which measures what proportion of the top-$N$ ranked $n$-grams match gold standard index terms. We are guaranteed coverage up to N=1000, as that is what was used to create the list of $n$-grams for annotation. We also use Mean Average Precision (MAP), a scalar version of the precision measurement, making it useful to evaluate choices of parameter settings or other operational decisions.

### 5.1 Index Term Results

The output of DP-seg is a nonoverlapping partitioning of the input corpus into a sequence of $n$-gram segments. For all our methods, we require an index term to be an $n$-gram segment of length $n \geq 2$ that is a noun chunk. Note that we start with a simplistic definition of a noun chunk: any $n$-gram that includes a sequence of adjectives or nouns ending in a noun. An initial indication of the segmented output from DP-seg can be seen in the segmentation of recent COLING and ACL article titles in Table 2. Based on cursory observation, many familiar bigrams (*dependency parsing*, *entity disambiguation*) appear in segments (note that we are showing the lemmatized and stopped text input). We also see some familiar longer segments (*loopy belief propagation*, *latent variable model*).

| Paper ID | DP-seg Title |
|---|---|
| C10-1002 | identify \| multiword expression \| leverage \| morphological \| syntactic idiosyncrasy |
| C10-1007 | fast accurate \| arc \| filtering \| dependency parsing |
| C10-1008 | hierarchical \| classifier \| applied \| multi-way \| sentiment detection |
| C10-1032 | entity disambiguation \| knowledge base population |
| C10-1034 | empirical study \| learning rank \| tweet |
| C10-1040 | emdc \| semi-supervised approach \| word alignment |
| P10-1005 | identifying \| generic \| noun phrases |
| P10-1006 | structural semantic relatedness \| knowledge-based method \| name entity disambiguation |
| P10-1007 | correct error \| speech recognition \| articulatory dynamics |
| P10-1025 | towards \| open-domain \| semantic role label |
| P10-1035 | accurate \| context-free parsing \| combinatory categorial grammar |
| P10-1038 | conditional random fields \| word \| hyphenation |
| P10-1040 | word representation \| simple \| general \| method \| semi-supervised learning |
| P10-1044 | latent dirichlet allocation \| method \| selectional preference |
| P11-1006 | fast accurate \| method \| approximate string search |
| P11-1017 | comprehensive \| dictionary \| multiword expression |
| P11-1026 | interactive \| topic modeling |
| P11-1027 | faster \| language models |
| P11-1033 | joint \| bilingual \| sentiment classification \| unlabeled \| parallel corpora |
| P11-1034 | pilot study \| opinion summarization \| conversation |
| P11-1039 | topical \| keyphrase extraction \| twitter |
| P11-1048 | comparison \| loopy belief propagation \| dual decomposition \| integrated \| ccg supertagging \| parsing |
| P11-1053 | semi-supervised relation extraction \| large-scale \| word clustering |

Table 2: Sample DP-seg of recent COLING and ACL paper titles (Paper ID is the ID from ACL Anthology).

| ACL | NIPS | DNA | HGT |
|---|---|---|---|
| training data (11924) | training data (1475) | gene expression (9822) | horizontal gene transfer (1371) |
| language model (6518) | data set (1300) | transcription factor (3557) | horizontal transfer (869) |
| test set (6233) | data point (1217) | gene expression profile (3536) | gene transfer (677) |
| noun phrase (5884) | loss function (1162) | dna microarray (3516) | escherichia coli (625) |
| natural language (5239) | training set (1133) | cell line (3462) | lateral gene transfer (586) |
| machine translation (4734) | objective function (1011) | microarray analysis (3429) | e. coli (524) |
| training set (4671) | covariance matrix (868) | breast cancer (2605) | phylogenetic analysis (434) |
| test data (4168) | generative model (867) | gene expression profiling (2313) | gene cluster (372) |
| data set (3920) | probability distribution (840) | microarray data (2147) | lateral transfer (283) |
| lexical item (3745) | optimization problem (838) | cdna microarray (1869) | evolutionary history (275) |
| target word (3703) | graphical model (822) | expression profile (1860) | antibiotic resistance (273) |
| lexical entry (3504) | special case (818) | target gene (1754) | genomic island (232) |

Table 3: Twelve most frequent types, corpus-wide.

Table 3 shows the 12 most frequent *n*-gram types from DP-seg, on our four corpora. The number in parentheses is the non-overlapping count of that exact type in the output of DP-seg (e.g. the 838 count of *optimization problem* for NIPS does not include any counts of *convex optimization problem*). The list of most frequent types in the table looks reasonable, given the domain of each corpus. Note that the most frequent types include some very general concepts (e.g. *training data* and *training set* are in both the ACL and NIPS top-12 list), and in the case of the PubMed search results, some unsurprising specific concepts, including our original query *dna microarray* and *horizontal gene transfer*. Note that the ACL and NIPS corpora only include bigrams in their top-12 list, reflecting the wider range of topics covered in these collections.

Table 4 and Table 5 show the 10 most frequent 3-gram and 4+ gram noun phrases from DP-seg for the ACL and NIPS corpora, with the number in parentheses again showing the count. We see that the 3-grams are approximately one order of magnitude less frequent than the bigrams shown in Table 3. We see another drop in the frequency for the 4+-grams, compared to the 3-grams. In the 4+-grams we see another phenomena: that of the definitional mention of a concept followed by the three (or more) letter acronym, such as *word sense disambiguation wsd*. In the original text, this would have appeared as *Word Sense Disambiguation (WSD)* with

| 3-grams | 4+ grams |
|---|---|
| natural language processing (1983) | natural language processing system (372) |
| word sense disambiguation (1838) | statistical machine translation system (348) |
| machine translation system (1463) | natural language processing nlp (346) |
| department computer science (1002) | word sense disambiguation wsd (338) |
| statistical machine translation (973) | support vector machine svm (260) |
| wall street journal (798) | statistical machine translation smt (238) |
| source target language (784) | noun verb adjective adverb (220) |
| semantic role labeling (768) | word error rate wer (207) |
| natural language processing (712) | natural language understanding system (199) |
| natural language generation (682) | latent semantic analysis lsa (172) |

Table 4: Ten most frequent 3-grams and 4+ grams for ACL.

| 3-grams | 4+ grams |
|---|---|
| support vector machine (230) | markov decision process mdp (93) |
| convex optimization problem (169) | support vector machine svm (87) |
| latent variable model (151) | markov chain monte carlo (86) |
| stochastic gradient descent (127) | latent dirichlet allocation lda (82) |
| high dimensional data (110) | markov chain monte carlo mcmc (67) |
| binary classification problem (103) | reproducing kernel hilbert space (62) |
| training test set (101) | principal component analysis pca (55) |
| number training example (101) | dirichlet process mixture model (47) |
| natural language processing (100) | conditional random field crf (45) |
| probability density function (97) | reproducing kernel hilbert space rkhs (42) |

Table 5: Ten most frequent 3-grams and 4+ grams for NIPS.

capitalization and parentheses around the acronym, which is lost in our preprocessing due to lowercasing and the removal of punctuation, including parentheses.

### 5.1.1 Empirical Evaluation

The P@$N$ results for DP-seg, c-value, nc-value and t-test are shown in Figure 1. The four panels correspond to each of our four corpora, with P@$N$ on the vertical axis, and $N$ on the horizontal axis. We see that DP-seg outperforms c-value, nc-value and t-test across the range of datasets and values of $N$. P@$N$ is a useful metric operationally, since one might consider or use the top-N ranked items from a system. We see the greatest difference of DP-seg against c/nc-value and t-test for the DNA and HGT datasets. This is largely due to the greater frequency of nested terminology, and the ability to DP-seg to better handle nested terminology.

When we examine the effectiveness over the top-ranked 3- and 4-grams and their subphrases in Figure 2, we see a consistent picture of DP-seg outperforming c-value and nc-value (note: there is no t-test curve here since t-test was only used to produce bigram index term candidates). The larger difference in results, particularly in the ACL and NIPS datasets, is due to the more challenging task of correctly solving the nesting problem. DP-seg is more adept at segmenting nested terms, e.g. it correctly identifies *statistical machine translation* and *machine translation* with high frequency, but *statistical machine* on its own is ranked quite low. Conversely, c/nc-value performs poorly on this task, incorrectly listing, e.g., *horizontal gene* as one of the top-ranked index terms.

This limitation of the c/nc-value algorithm is highlighted in Table 6, which compares the top-ranked index terms from DP-seg and nc-value on the HGT dataset. We see that DP-seg has 100% precision over the top-12 returned results, while nc-value only has 75% precision. Despite its attempt to properly discount nested subphrases, c/nc-value fails to recognize that terms such

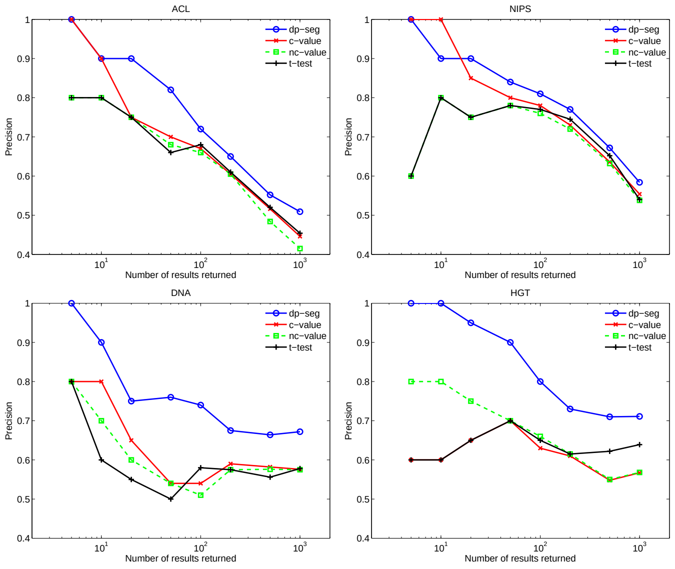

Figure 1: P@$N$ after N results returned, for N up to top-1000 ranked results.

as *horizontal gene* do not exist.

### 5.1.2 Effect of Model Parameters

The DP-seg model has only two parameters: $\alpha$ and $p_\#$. $\alpha$ controls sparsity, i.e. the number of distinct $n$-gram types. The probability of segment boundary $p_\#$ controls the average length of $n$-grams. In our results to date, we have used the default values of $\alpha = 20$ and $p_\# = 0.5$.

We observe the effect of $\alpha$ and $p_\#$ using the NIPS and HGT datasets in Figure 3. The left panel shows the size of the $n$-gram lexicon versus $\alpha$. As $\alpha$ increases eight orders-of-magnitude, we see approximately a one order-of-magnitude increase in the size of the multiword lexicon. The right panel shows the percentiles of $n$-gram lengths versus $p_\#$. As $p_\#$ decreases from 0.999 to 0.001 (viewing right to left), we see an increase in the length of the longest $n$-grams. Note that around our operating range (i.e. $p_\# = 0.5$), the curves produce a reasonable distribution of $n$-gram lengths. The average length of $n$-gram segments has a relatively weak dependence on $p_\#$. As $p_\#$ increases from 0.001 to 0.999 (arguably relatively extreme values), the average length decreases only from 1.34 to 1.30 (NIPS), and decreases from 1.28 to 1.24 (HGT).

The effect of $\alpha$ and $p_\#$ on Mean Average Precision (MAP) for NIPS and HGT is shown in Figure 4. We see a slight decrease in MAP as the DP concentration parameter $\alpha$ increases, although the reduction in MAP is slight compared to the large increase in $\alpha$. As the probability of segment boundary $p_\#$ increases from 0.1 to 0.9, there is almost no effect on MAP. The rel-

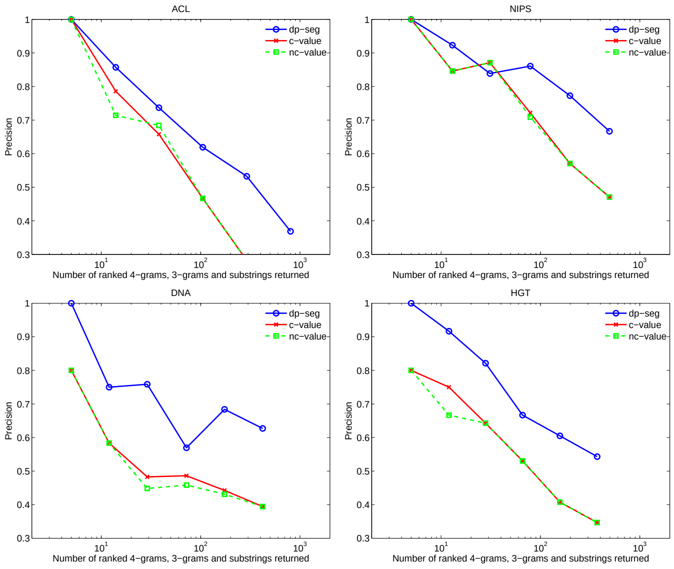

Figure 2: Precision for top-100 4-grams, 3-grams, and subphrases.

atively weak effect of the parameters on MAP is explained by MAP being a precision-focused metric, and that we only have relatively high frequency positive examples labeled. The precision/recall tradeoff is well known for this type of parameter, with precision decreasing and recall increasing as $\alpha$ increases.

Given our Bayesian framework, we could in principle learn $\alpha$ and $p_{\#}$, however we opt for the simpler choice of fixing $\alpha$ and $p_{\#}$, and leave this optimization for future work.

### 5.1.3 The Effects of Pre-Processing: POS Tagging and Stopword Removal

One advantage of DP-seg is that it can run on any stream of tokens, and does not require lemmatization, POS tagging, parsing or removal of common stopwords (stopping). To allow for a fair comparison with c/nc-value algorithm, we did POS tag and remove stopwords for datasets in Section 4, creating a single input representation used by all algorithms (including t-test). In this section we investigate the effect of relaxing this preprocessing for DP-seg, and running DP-seg without POS tags, and without removing stopwords.

The Mean Average Precision (MAP) results for HGT and NIPS are shown in Table 7a. The first row shows our baseline case of POS tagged and stopped. In the second row, DP-seg is run without POS tags. Since our gold standard only contains noun phrases, we only evaluate using noun phrases. In the third row, DP-seg is run on raw text that only has punctuation removed. Again the final output is filtered for noun phrases. We see a relatively robust result that the performance of DP-seg is relatively good, even with minimal preprocessing.

| DP-seg | nc-value |
|---|---|
| horizontal gene transfer | gene transfer |
| horizontal transfer | horizontal gene transfer |
| gene transfer | *horizontal gene** |
| escherichia coli | lateral gene transfer |
| lateral gene transfer | horizontal transfer |
| e. coli | *lateral gene** |
| phylogenetic analysis | e. coli |
| gene cluster | escherichia coli |
| lateral transfer | phylogenetic analysis |
| evolutionary history | antibiotic resistance |
| antibiotic resistance | *resistance gene** |
| genomic island | gene cluster |

Table 6: Comparison of top-12 ranked index terms from DP-seg and nc-value on the HGT dataset (incorrect terms are italicized and marked with *).

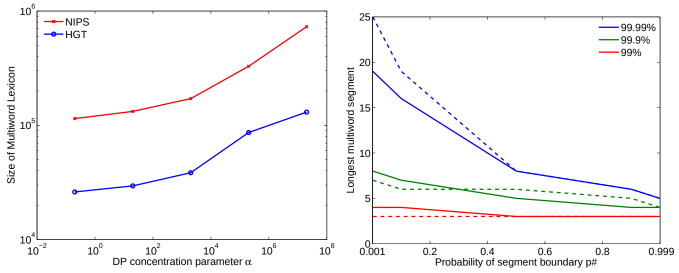

Figure 3: Effect of model parameters $\alpha$ and $p_{\#}$ on the lexicon size and distribution of $n$-gram length (in the right-hand figure, solid lines are percentiles from NIPS, and dashed lines are percentiles from HGT).

In the SemEval experiments in Section 5.2, DP-seg is run on the provided text as is, without any POS tagging or stopword removal.

## 5.2 SemEval Results

We ran DP-seg on the SemEval corpus of 244 fulltext articles. We used the input text as is (which was stemmed), for consistency with the published SemEval results. We performed the same evaluation as Kim et al. (2010), computing micro-averaged F-scores calculated over the top-5/10/15 candidates, with a candidate only being considered a match if there is an exact match with one of the gold standard keyphrases. We base the selection of the top-$N$ candidates on only $n$-grams, ordered by the token-level frequency within the document.

The results are shown in Table 8, with the rows ordered by overall performance. DP-seg (which is completely unsupervised) easily outperforms the unsupervised TF-IDF-based baseline system. In all cases except one, DP-seg outperforms the average score across twenty special-purpose built systems for keyphrase extraction (second row). If DP-seg were competing in this shared task, it would have ranked 11th out of twenty systems based on the top-15 results (and 9th based on the top-5 and top-10 results) and beaten the average for participating systems in all of top-5, top-10 and top-15 evaluations (Kim et al., pear). Note, however, that only two of the systems that performed better than DP-seg are unsupervised (namely El-Beltagy and Rafea (2010) and Bordea and Buitelaar (2010)); all other systems are explicitly trained on the train-

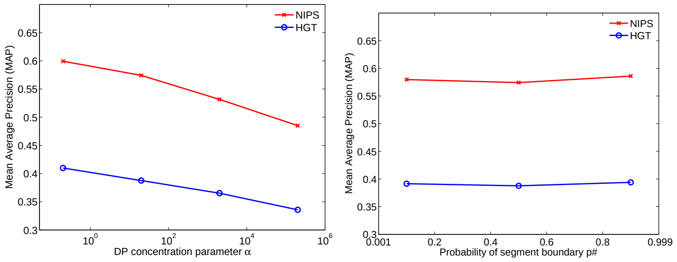

Figure 4: Effect of model parameters $\alpha$ and $p_\#$ on Mean Average Precision

| Processing | HGT | NIPS |
|---|---|---|
| POS tagged & Stopped | 0.57 | 0.40 |
| not POS tagged & Stopped | 0.51 | 0.43 |
| not POS tagged & not Stopped | 0.55 | 0.36 |

(a) HGT and NIPS Mean Average Precision scores for various pre-processing strategies.

| **MWE** |
|---|
| *horizontal gene transfer* (1161) |
| *horizontal gene transfer* hgt (249) |
| by *horizontal gene transfer* (176) |
| *horizontal gene transfer* and (133) |
| *horizontal gene transfer* event (132) |
| *horizontal gene transfer* in (90) |
| via *horizontal gene transfer* (71) |
| through *horizontal gene transfer* (71) |

(b) Effect of not removing stopwords. List of of frequent segments that include “horizontal gene transfer”.

Table 7: Effect of POS tagging and stopword removal.

ing documents provided for the task. The fact that we are competitive over the task without any tweaking of the method or reranking of the results is highly encouraging, and complements the P@$N$ results in Section 5.1.1. Also note that this experiment was run without POS tagging and without stopwords removed, adding to the evidence that DP-seg is robust to these preprocessing choices.

## 6 Discussion and Conclusion

Our results indicate that DP-seg performs well at both index term identification and keyphrase extraction. As part of this, it performs well at disambiguating nested index terms from scientific literature — that is determining an optimal non-overlapping partitioning of a stream of words in a corpus — into a sequence of $n$-grams. Places where DP-seg tends not to perform so well are examples such as *previous work* and boilerplate text such as authors thanking *anonymous reviewers [for] helpful comments*, which are formulaic and certainly characteristic of the domains we are working with, but not appropriate index terms of keyphrases. Given that DP-seg is based simply on the text stream and doesn’t capture positional information in the document, it has no way of differentiating these from conceptually-grounding index terms. One possible research direction for dealing with this issue would be to incorporate positional information (e.g. document zones or argumentative zoning (Teufel et al., 1999)).

As Goldwater et al. found, the unigram DP-seg model tends to be overconfident with very limited data. If it has never seen a particular $n$-gram, it is reluctant to form that $n$-gram, but

| Method | Top-5 | | | Top-10 | | | Top-15 | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| Baseline | 22.0 | 7.5 | 11.2 | 17.7 | 12.1 | 14.4 | 14.9 | 15.3 | 15.1 |
| Avg System | 28.7 | 9.8 | 14.6 | 23.2 | 15.8 | 18.8 | 19.9 | 20.4 | 20.1 |
| DP-seg | 33.3 | 10.5 | 16.0 | 23.7 | 16.9 | 19.7 | 19.2 | 21.3 | 20.2 |
| Best System | 39.0 | 13.3 | 19.8 | 32.0 | 21.8 | 26.0 | 27.2 | 27.8 | 27.5 |

Table 8: Precision, Recall, and F-score on the SemEval dataset.

if it has seen just one occurrence, it is very likely to produce more of the same *n*-gram. In response to this, Goldwater proposed a bigram model that includes a Hierarchical Dirichlet Process (HDP) in place of the Dirichlet Process. We plan to investigate the performance of the Bigram/HDP model in future work.

One aspect of index term identification which we have ignored in this research is the location of the index term in the document collection, largely because of the much higher annotation complexity. If we are to deliver on the promise of building an automatic index generator, we need to additionally determine which instances of a given index term should be indexed. For example, for a paper which mentions *support vector machines* on every page, should each page be indexed, just the first mention, or a defining mention (e.g. in a section describing the methodology)? Related to this question is the question of index term equivalence. For example, if the paper mentions both *support vector machines* and *SVMs*, we would ideally like to be able to recognize them as interchangeable and alternative forms of the same index term. Again, the current model is unable to pick up on such equivalences, but a fully-functioned index term identification system should be able to deal with cases such as this, suggesting a direction for future work. Another possilble extension is the determination of noun chunk boundaries, which is currently based on a simple POS sequence heuristic. In terms of precision, the heuristic is robust and not problematic, but in terms of recall, we potentially miss more complex term candidates. Using a noun phrase chunker or parser could be used to boost recall.

An interesting observation is the difficulty c/nc-value has in correctly extracting highly ranked index terms for the HGT dataset, as was shown in Table 6. We believe that, while c/nc-value is designed to discount shorter *n*-grams that appear as subphrases in longer *n*-grams, the wide variety of contexts makes this a hard problem, and that there is no obvious fix to c/nc-value to solve this problem. We argue that the Bayesian segmentation and generative model from DP-seg provides a more principled solution to this problem — and one that performs significantly better. The Bayesian framework also allows for other extensions to incorporate additional lexical or syntactic data into the model. We leave these investigations for future work.

To summarize, we have applied an unsupervised Bayesian method to the tasks of index term identification and keyphrase extraction. Index term identification is the task of automatically determining terms to include in a literary index for a document collection, and we constructed a total of four datasets across a range of scientific domains. Keyphrase extraction is defined in the conventional way, and was evaluated relative to the SemEval-2010 dataset. We found the Bayesian method to outperform both a multiword terminology extraction method and a collocation extraction baseline over the index term identification task, and to rank above the average results for participating systems in the SemEval-2010 task, beaten by only two unsupervised systems (in addition to a number of supervised systems), without any tuning to the task.

## Acknowledgements

DN supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center contract number D11PC20155. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government. DN also supported by NSF-1106434. NICTA is funded by the Australian government as represented by Department of Broadband, Communication and Digital Economy, and the Australian Research Council through the ICT centre of Excellence programme.

## References

Baldwin, T. and Kim, S. N. (2009). Multiword expressions. In Indurkhya, N. and Damerau, F. J., editors, *Handbook of Natural Language Processing*. CRC Press, Boca Raton, USA, 2nd edition.

Bordea, G. and Buitelaar, P. (2010). Deriunlp: A context based approach to automatic keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 146–149, Uppsala, Sweden.

El-Beltagy, S. R. and Rafea, A. (2010). Kp-miner: Participation in semeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 190–193, Uppsala, Sweden.

Evert, S. (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD thesis, University of Stuttgart.

Frantzi, K., Ananiadou, S., and Mima, H. (2000). Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3:115–130.

Goldwater, S., Griffiths, T., and Johnson, M. (2005). Interpolating between types and tokens by estimating power-law generators. In *NIPS*.

Goldwater, S., Griffiths, T., and Johnson, M. (2009). A bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112:21–54.

Gutwin, C., Paynter, G., Witten, I., Nevill-Manning, C., and Frank, E. (1999). Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1-2):81–104.

Kim, S. and Baldwin, T. (2010). How to pick out token instances of English verb-particle constructions. *Language Resources and Evaluation*, 44(1-2):97–113.

Kim, S. N., Medelyan, O., Kan, M., and Baldwin, T. (2010). Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden.

Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. (to appear). Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*.

Leonard, L. (1977). Inter-indexer consistency studies, 1954-1975: a review of the literature and summary of study results. Technical report, Graduate School of Library and Information Science. University of Illinois at Urbana-Champaign.

Pecina, P. (2008). *Lexical Association Measures*. PhD thesis, Charles University.

Radev, D. R., Muthukrishnan, P., and Qazvinian, V. (2009). The ACL anthology network corpus. In *Proceedings, ACL Workshop on Natural Language Processing and Information Retrieval for Digital Libraries*, Singapore.

Teufel, S., Carletta, J., and Moens, M. (1999). An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 110–117, Bergen, Norway.

Witten, I., Paynter, G., Frank, E., Gutwin, C., and Nevill-Manning, C. (1999). Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries (DL-1999)*, pages 254–255, Berkeley, California.