Automatic Labelling of Topics with Neural Embeddings

Shraey Bhatia[1,2], **Jey Han Lau**[1,2] and **Timothy Baldwin**[2]
[1] IBM Research
[2] Dept of Computing and Information Systems,
The University of Melbourne
shraeybhatia@gmail.com, jeyhan.lau@gmail.com, tb@ldwin.net

Abstract

Topics generated by topic models are typically represented as list of terms. To reduce the cognitive overhead of interpreting these topics for end-users, we propose labelling a topic with a succinct phrase that summarises its theme or idea. Using Wikipedia document titles as label candidates, we compute neural embeddings for documents and words to select the most relevant labels for topics. Compared to a state-of-the-art topic labelling system, our methodology is simpler, more efficient, and finds better topic labels.

1 Introduction

Topic models are a popular approach to detecting trends and traits in document collections, e.g. in tracing the evolution of a scientific field through its publications (Hall et al., 2008), enabling visual navigation over search results (Newman et al., 2010a), interactively labelling document collections (Poursabzi-Sangdeh et al., 2016), or detecting trends in text streams (Wang and McCallum, 2006; AlSumait et al., 2008). They are typically unsupervised, and generate "topics" t_i in the form of multinominal distributions over the terms w_j of the document collection ($\Pr(w_j|t_i)$), and topic distributions for each document d_k in the collection, in the form of a multinomial distribution over topics ($\Pr(t_i|d_k)$). Traditionally, this has been carried out based on latent Dirichlet allocation (LDA: Blei et al. (2003)) or extensions thereof, but more recently there has been interest in deep learning approaches to topic modelling (Cao et al., 2015; Larochelle and Lauly, 2012).

In contexts where the output of the topic model is presented to a human user, a fundamental concern is the best way of presenting the rich information generated by the topic model, in particular, the topics themselves, which provide the primary insights into the document collection. The de facto topic representation has been a simple term list, in the form of the top-10 terms in a given topic, ranked in descending order of $\Pr(w_j|t_i)$. The cognitive overhead in interpreting the topic presented as a list of terms can be high, and has led to interest in the task of generating labels for topics, e.g. in the form of textual descriptions (Mei et al., 2007; Lau et al., 2011; Kou et al., 2015), visual representations of the topic words (Smith et al., to appear), or images (Aletras and Stevenson, 2013). In the former case, for example, rather than the top-10 terms of ⟨*school, student, university, college, teacher, class, education, learn, high, program* ⟩, a possible textual label could be simply EDUCATION. Recent work has shown that, in the context of a timed information retrieval (IR) task, automatically-generated textual labels are easier for humans to interpret than the top-10 terms, and lead to equivalent-quality relevance judgements (Aletras et al., 2014). Despite this, the accuracy of state-of-the-art topic generation methods is far from perfect, providing the motivation for this work.

In this paper, we propose an approach to topic labelling based on word and document embeddings, which both automatically generates label candidates given a topic input, and ranks the candidates in either an unsupervised or supervised manner, to produce the final topic label. Our contributions in this work are: (1) a label generation approach based on combined word and document embeddings, which is both considerably simpler than and empirically superior to the state-of-the-art generation method; (2) a

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 953–963, Osaka, Japan, December 11-17 2016.

simple label ranking approach that exploits character and lexical information, which is also superior to the state-of-the-art ranking approach; and (3) release of an open source implementation of our method, including a new dataset for topic label ranking evaluation.[1]

2 Related Work

Mei et al. (2007) introduced the task of generating labels for LDA topics, based on first extracting bigram collocations from the topic-modelled document collection using a lexical association measure, and then ranking them based on KL divergence with each topic. The approach is completely unsupervised.

Lau et al. (2011) proposed using English Wikipedia to automatically label topics. First, they map the topic to a set of concepts by querying Wikipedia using the top-10 topic terms based on: (a) Wikipedia's native search API; and (b) Google's search API, with site restriction. The top-8 article titles from each of these two sources are pooled to generate the primary candidate topic labels. Secondary labels are generated from component n-grams contained within the primary candidates, and filtering out incoherent and unrelated titles using the `RACO` measure (Grieser et al., 2011) to measure similarity with the primary labels, based on Wikipedia document categories. The combined set of primary and secondary label candidates is then ranked using a number of lexical association features, either directly in an unsupervised manner, or indirectly based on training a support vector regression model. The authors provide an extensive analysis of their method with that of Mei et al. (2007), and find their label generation and ranking methodology to be empirically superior (in both an unsupervised and supervised setting). In this paper, we seek to improve upon the topic labelling benchmark set by Lau et al. (2011).

Hulpus et al. (2013) developed a graph-based method for topic labelling, leveraging the structure of DBpedia concepts. Their approach is styled around graph-based word sense disambiguation, and extracts a set of DBpedia concepts corresponding to the top-N terms of a topic. They then construct a graph centred around DBpedia concepts and filter noise based on graph connectivity (the hypothesis being that sense graphs of words from a topic should be connected). To find the best label for a topic, they experiment with a variety of graph centrality measures.

In work slightly further afield, Zhao et al. (2011) proposed topical keyphrase extraction for Twitter. Although the work focuses mainly on Twitter, the methodology can be applied to other domains and to label topics. Zhao et al. (2011) follow a three-step process for keyphrase extraction: (1) keyword ranking; (2) candidate keyphrase generation (based on the individual keywords); and (3) keyphrase ranking. They use a novel topic context-sensitive *PageRank* method to regularise topic scores for keyword ranking, and a probabilistic scoring method that takes into account relevance, interestingness and keyphrase length for keyphrase ranking.

Not directly for the purposes of topic labelling but as a relevant pretraining method, Mikolov et al. (2013) proposed `word2vec` to learn word embeddings, which they found to perform strongly over a range of word similarity tasks, and also to be useful for initialising deep learning models. Two approaches are proposed in the paper: `cbow` and `skip-gram`. `cbow` combines neighbouring words to predict a target word, while `skip-gram` uses the target word to predict neighbouring words. Both approaches use a feedforward neural network with a non-linear hidden layer to maximize the objective function; to improve computational efficiency, the authors propose using negative sampling. In this paper, we will use `word2vec` as a means of generating topic term and label representations.

As an extension of `word2vec`, Le and Mikolov (2014) introduced `doc2vec` to learn embeddings for word sequences (e.g. paragraphs or documents). By treating each document as a word token, the same `word2vec` methodology can be used to learn document embeddings. The authors propose two implementations: `dbow`, which uses the document vector to predict its document words, and is the `doc2vec` equivalent of `skip-gram`; and `dmpv`, which uses a small window of words and concatenates them with the document vector to predict a document word, and is the `doc2vec` equivalent of `cbow`.[2] Compared to `dbow`, `dmpv` takes into account the local word ordering, and has a higher number of parameters, since the input is a

[1] `https://github.com/sb1992/NETL-Automatic-Topic-Labelling-`

[2] Strictly speaking, `cbow` combines word vectors by summing them, while `dmpv` combine word vectors and document vector by concatenating them.

concatenation of vectors. As with word2vec, we will use doc2vec as an alternative means of generating topic term and label representations.

Building on word2vec, Kou et al. (2015) experimented with neural embeddings in the context of topic labelling. In addition to skip-gram and cbow word vectors, the authors also included letter trigram vectors of a word, with the rationale that it generalises over morphologically-related forms of the same word. Their methodology consists of first generating candidate labels for topics from topic-related documents using a chunk parser. By representing both topic words and topic labels using word embeddings and letter trigrams, they rank the labels using cosine similarity to obtain the best label for a topic. In their evaluation, they find that simple letter trigrams are ultimately the most reliable means of label ranking.

3 Methodology

Following Lau et al. (2011), our method is made up of two steps: (1) topic label generation based on English Wikipedia; and (2) topic label ranking, based on a supervised learn-to-rank model. We detail each of these steps below.

3.1 Candidate Generation

To match topics to Wikipedia articles,[3] Lau et al. (2011) used an IR approach, by querying English Wikipedia with the top-N topic terms. However, in order to do this, they required external resources (two search APIs, one of which is no longer publicly available), limiting the general-purpose utility of the method. We propose an alternative approach: precomputing distributed representations of the topic terms and article titles using word2vec and doc2vec.

To this end, we train a doc2vec model on the English Wikipedia articles, and represent the embedding of a Wikipedia title by its document embedding. As doc2vec runs word2vec internally, word embeddings are also learnt during the training. Given the top-N topic terms, the topic embedding is represented by these terms' word embeddings. Based on the findings of Lau and Baldwin (2016) that the simpler dbow has less parameters, trains faster, and performs better than dmpv in several extrinsic tasks, we experiment only with dbow.[4] In terms of hyper-parameter settings, we follow the recommendations of Lau and Baldwin (2016).[5]

In addition to doc2vec, we also experiment with word2vec to generate embeddings for Wikipedia titles. By treating titles as a single token (e.g. concatenating *financial crisis* into *financial_crisis*) and greedily tokenising the text of all of the Wikipedia articles, we can then generate word embeddings for the titles. For word2vec, we use the skip-gram implementation exclusively.[6]

For both doc2vec and word2vec, we first pre-process English Wikipedia,[7] using Wiki Extractor to clean and extract Wikipedia articles from the original dump.[8] We then tokenise words with the Stanford CoreNLP Parser (Klein and Manning, 2003), and lowercase all words. We additionally filter out articles where the article body is made up of less than 40 words, and also disambiguation pages. We also remove titles whose length is longer than 4 words, as they are often too specific or inappropriate as topic labels (e.g. *List of Presidents of the United States*). For word2vec, we remove any parenthesised sub-component of an article title — e.g. in the case of *Democratic Party (United States)*, we remove *(United States)* to generate *Democratic Party* — as we would not expect to find verbatim usages of the full title. This has the potential side-effect of mapping multiple articles onto a single ambiguous title, resulting in multiple representations for *Democratic Party*. While acknowledging that there are instances where the more specific title may be appropriate as a label, the generalised version is always going to be a hypernym of the original, and thus appropriate as a label candidate.

[3] As of 2016 there are over 5 million documents in English Wikipedia.

[4] We use Gensim's implementation of both doc2vec and word2vec for all experiments: https://radimrehurek.com/gensim/.

[5] doc2vec hyper-parameters: sub-sampling threshold $= 10^{-5}$, vector size $= 300$, window size $= 15$, negative sample size $= 5$, and training epochs $= 20$.

[6] word2vec hyper-parameters: sub-sampling threshold $= 10^{-5}$, vector size $= 300$, window size $= 5$, negative sample size $= 5$, and training epochs $= 100$.

[7] The English Wikipedia dump used in all experiments is dated 2015-12-01.

[8] https://github.com/attardi/wikiextractor/

Top-10 Topic Terms	word2vec Labels	doc2vec Labels
	software	microsoft visual studio
blogs, vmware, server, virtual, oracle,	desktop	desktop virtualization
update, virtualization, application,	operating system	microsoft exchange server
infrastructure, management	virtualization	cloud computing
	middleware	windows server 2008

Table 1: The top-5 labels generated using doc2vec and word2vec title embeddings for the topic provided

Given a topic, we measure the relevance of each title embedding (generated by either doc2vec or word2vec) based on the pairwise cosine similarity with each of the word embeddings for the top-10 topic terms, and aggregate by taking the arithmetic mean. Formally, the doc2vec relevance (rel_{d2v}) and word2vec relevance (rel_{w2v}) of a title a and a topic T is given as follows:

$$rel_{d2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{d}_{d2v}(a), E^{w}_{d2v}(v)\right) \quad (1)$$

$$rel_{w2v}(a,T) = \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{w}_{w2v}(a), E^{w}_{w2v}(v)\right) \quad (2)$$

where $E^{d}_{d2v}(x)$ is the document embedding of title x generated by doc2vec; $E^{w}_{d2v}(y)$ is the word embedding of word y generated by doc2vec; $E^{w}_{w2v}(z)$ is the word embedding of word z generated by word2vec; $v \in T$ is a topic term; $|T|$ is the number of topic terms (10 in our experiments); and $\cos(\vec{x},\vec{y})$ is the cosine similarity function.

The idea behind using both doc2vec and word2vec to generate title embeddings is that we observe that the two models favour different types of labels: doc2vec tends to favour fine-grained concepts, while word2vec favours more generic or abstract labels. As an illustration of this, in Table 1 we present one of the actual topics used later in our evaluation, and the top-5 article titles based on doc2vec and word2vec. This dichotomy is rooted in the differences in the modelling of context in the two models. In doc2vec, the title embedding is determined by the words that belong to the title, each of which is in turn determined by its context of use; it thus directly captures the compositional semantics of the title. With our word2vec method, on the other hand, the title embedding is determined directly by the neighbouring words of the title token in text, oblivious to the composition of words in the title.

To combine the strengths of doc2vec and word2vec, for each topic we generate a combined candidate ranking by summing the relevance scores using top-100 candidates from doc2vec and word2vec:[9]

$$rel_{d2v+w2v}(a,T) = rel_{d2v}(a,T) + rel_{w2v}(a,T) \quad (3)$$

3.2 Candidate Ranking

The next step after candidate generation is to re-rank them based on a supervised learn-to-rank model, in an attempt to improve the quality of the top-ranking candidates.

The first feature used in the supervised reranker is *LetterTrigram*, and based on the finding of Kou et al. (2015) that letter trigram vectors are an effective means of ranking topic labels. Our implementation of their method is based on measuring the overlap of letter trigrams between a given topic label and the topic words. For each topic, we first convert each topic label and topic words into multinomial distributions over letter trigrams, based on simple maximum likelihood estimation.[10] We then rank the

[9] From preliminary experiments we found that summing only the top-100 candidates from doc2vec and word2vec is better than summing all candidates. As we remove the parenthesised sub-component of an article title for word2vec (*Democratic Party (United States)* $\longrightarrow$ *Democratic Party*) , we observe that these titles tend to be very general and can ocassionally produce very high cosine similarity and skew the combined score for a number of similar labels (e.g. causing *Democratic Party* from a host of countries (*Democratic Party (United States)*, *Democratic Party (Australia)*, etc) to appear in the top ranking).

[10] For topic words, the letter trigrams are generated by parsing each of the topic words as separate strings rather than one concatenated string.

labels based on their cosine similarity with the topic words. The rank value constitutes the first feature of the supervised learn-to-rank model. Additionally, this ranking by letter trigram method also forms our unsupervised baseline, as we found that it to have the best unsupervised ranking performance of all our features, consistent with the findings of Kou et al. (2015).[11]

The second feature is *PageRank* (Page et al., 1998), in an attempt to prefer labels which represent more "core" concepts in Wikipedia. *PageRank* uses directed links to estimate the significance of a document, based on the probability of a random web surfer visiting a web page by either following hyperlinks or randomly transporting to a new page. We construct a directed graph from Wikipedia based on hyperlinks within the article text, and from this, compute a *PageRank* value for each Wikipedia article (and hence, title).[12]

Our last two features are lexical features proposed by Lau et al. (2011): (1) *NumWords*, which is simply the number of words in the candidate label (e.g. *operating system* has 2 words); and (2) *TopicOverlap*, which is the lexical overlap between the candidate label and the top-10 topic terms (e.g. *desktop virtualization* has a *TopicOverlap* score of 1 in our example from Table 1).

Given these features and a gold standard order of candidates (detailed in Section 4.1), we train a support vector regression model (SVR: Joachims (2006)) over these four features.

4 Datasets

For direct comparison with Lau et al. (2011), we use the same set of topics they used in their experiments. These were generated from 4 different domains: BLOGS, BOOKS, NEWS and PUBMED. In general, BLOGS, BOOKS and NEWS cover wide-ranging topics from product reviews to religion to finance and entertainment, whereas PubMed is medical-domain specific.

BLOGS is made up of 120k blog articles from the Spinn3r blog dataset; BOOKS is made up of 1k English language books from the Internet Archive American Libraries collection; NEWS is made up of 29k New York Times articles from English Gigaword; and PUBMED is made up of 77k PubMed biomedical abstracts. Lau et al. (2011) ran LDA on these documents and generated 100 topics for each domain. They filtered incoherent topics using an automated approach (Newman et al., 2010b), resulting in 45, 38, 60, 85 topics for BLOGS, BOOKS, NEWS and PUBMED, respectively.

4.1 Gold Standard Judgements

To evaluate our method and train the supervised model, gold-standard ratings of the candidates are required. To this end, we used CrowdFlower to collect human judgements.[13] We follow the approach of Lau et al. (2011), presenting 10 pairings of topic and candidate label, and asking human judges to rate the label on an ordinal scale of 0–3 where 0 indicates a completely inappropriate label, and 3 indicates a very good label for the given topic.

To control for annotation quality, we make use of the original annotations released by Lau et al. (2011). We select labels with a mean rating ≥ 2.5 (good labels) and ≤ 0.5 (bad labels) to serve as controls in our tasks. We include an additional topic–label control pair in addition to the 10 topic–label pairs in a HIT. Control pairs are selected randomly without replacement, and randomly injected into the HIT. To pass quality control, a worker is required to rate bad labels ≤ 1.0 and good labels ≥ 2.0. A worker is filtered out if his/her overall pass rate over all control pairs is < 0.75.

Each candidate label was rated by 10 annotators. Post-filtered, we have an average of 6.4 annotations for each candidate label.[14] To aggregate the ratings for a candidate label, we compute its mean rating, and rank the candidate labels based on the mean ratings to produce the gold standard ranking for each topic.

[11] Note that we do not make use of the noun chunk-based label generation methodology of Kou et al. (2015), in line with the findings of Lau et al. (2011) that Wikipedia titles give rise to better label candidates than n-grams extracted from the topic-modelled documents.

[12] We use the following implementation for *PageRank*: `https://www.nayuki.io/page/computing-wikipedias-internal-pageranks/`

[13] `https://www.crowdflower.com/`

[14] Post-filering, some candidates ended up with less than 3 annotations; these candidates were posted for another annotation round to gather more annotations.

Test Domain	Training	Top-1 Avg.		nDCG-1		nDCG-3		nDCG-5	
		LGNB	NETL	LGNB	NETL	LGNB	NETL	LGNB	NETL
BLOGS	Baseline	1.84	**1.91**	0.75	**0.77**	0.77	**0.82**	0.79	**0.83**
	In-Domain	1.98	**2.00**	0.81	0.81	0.82	**0.85**	0.83	0.84
	Cross-domain: BOOKS	1.88	**1.91**	0.77	0.78	0.81	**0.83**	0.83	0.83
	Cross-domain: NEWS	**1.97**	1.92	**0.80**	0.78	0.83	0.84	0.83	0.84
	Cross-domain: PUBMED	**1.95**	1.90	**0.80**	0.77	0.82	0.83	0.83	0.83
	Cross-domain: All 3	—	1.92	—	0.78	—	0.84	—	0.84
	Upper Bound	2.45	**2.48**	1.00	1.00	1.00	1.00	1.00	1.00
BOOKS	Baseline	1.75	**1.97**	0.77	0.78	0.77	**0.82**	0.79	**0.83**
	In-Domain	1.91	**1.99**	**0.84**	0.82	0.81	0.82	0.83	0.84
	Cross-domain: BLOGS	1.82	**2.02**	0.79	**0.83**	0.81	0.82	0.82	**0.84**
	Cross-domain: NEWS	1.82	**1.99**	0.79	**0.81**	0.81	0.82	0.83	0.84
	Cross-domain: PUBMED	1.87	**1.97**	0.81	0.80	0.82	0.82	0.83	0.84
	Cross-domain: All 3	—	2.03	—	0.83	—	0.83	—	0.84
	Upper Bound	2.29	**2.49**	1.00	1.00	1.00	1.00	1.00	1.00
NEWS	Baseline	1.96	**2.04**	0.80	**0.82**	0.79	**0.84**	0.78	**0.85**
	In-Domain	2.02	2.02	**0.82**	0.80	0.82	**0.84**	0.84	0.85
	Cross-domain: BLOGS	2.03	2.03	**0.83**	0.81	0.82	**0.84**	0.84	0.85
	Cross-domain: BOOKS	**2.01**	1.98	**0.82**	0.79	0.82	0.83	0.83	0.84
	Cross-domain: PUBMED	2.01	2.00	**0.82**	0.79	0.82	0.83	0.83	0.84
	Cross-domain: All 3	—	1.99	—	0.79	—	0.84	—	0.84
	Upper Bound	2.45	**2.56**	1.00	1.00	1.00	1.00	1.00	1.00
PUBMED	Baseline	1.73	**1.94**	0.75	**0.79**	0.77	**0.80**	0.79	**0.82**
	In-Domain	1.79	**1.99**	0.77	**0.81**	0.82	0.81	**0.84**	0.82
	Cross-domain: BLOGS	1.80	**1.98**	0.78	**0.80**	0.82	0.81	**0.84**	0.82
	Cross-domain: BOOKS	1.77	**1.98**	0.77	**0.80**	0.82	0.81	0.83	0.82
	Cross-domain: NEWS	1.79	**1.98**	0.77	**0.80**	0.82	0.81	**0.84**	0.82
	Cross-domain: All 3	—	2.01	—	0.81	—	0.81	—	0.82
	Upper Bound	2.31	**2.51**	1.00	1.00	1.00	1.00	1.00	1.00

Table 2: Results across the four domains. Boldface indicates the better system between NETL and LGNB (with an absolute difference > 0.01).

We collect judgements for the top-19 candidates from the unsupervised ranking.[15] For candidate ranking (Section 3.2), we are therefore re-ranking the top-19 candidates.

5 Experiments

In this section we present the results of our topic labelling experiments, and compare our method with that of Lau et al. (2011). Henceforth we refer to our method as "NETL" (neural embedding topic labelling), and Lau et al. (2011) as "LGNB".

Following LGNB, we use **top-1 average rating** and **normalized discounted cumulative gain (nDCG)** (Järvelin and Kekäläinen, 2002; Croft et al., 2009) as our evaluation metrics. Top-1 average computes the mean rating of the top-ranked labels, and provides an evaluation of the absolute utility of the preferred labels. nDCG, on the other hand, measures the relative quality of the ranking, calibrated relative to the ratings of the gold-standard ranking. Similarly to LGNB, we compute nDCG for the top-1 (nDCG-1), top-3 (nDCG-3), and top-5 (nDCG-5) ranked labels.

5.1 Results

Following LGNB, we present results for: (a) the unsupervised ranker (based on letter trigrams); (b) the supervised re-ranker in-domain, based on 10-fold cross validation, averaged over 10 runs with different

[15] Ideally we would have liked to have collected judgements for as many candidates as possible, but due to budget constraints we were only able to have the top-19 annotated.

Domain	Topic Terms	Label Candidate
BLOGS	vmware server virtual oracle update virtualisation application infrastructure management microsoft	virtualisation
BOOKS	church archway building window gothic nave side value tower	church architecture
NEWS	investigation fbi official department federal agent investigator charge attorney evidence	criminal investigation
PUBMED	rate population prevalence study incidence datum increase mortality age death	mortality rate

Table 3: A sample of topics and their topic labels generated by NETL.

partitionings; (c) the supervised re-ranker cross-domain; and (d) the upper bound, based on a perfect ranking of the candidates. For cross-domain learning, we train our model using one domain and test it on a different domain, or alternatively combine data from three domains and test on the remaining fourth domain, e.g. training on BOOKS +NEWS +PUBMED and testing on BLOGS. Cross-domain results give us a more accurate picture of the performance of our methodology in real-world applications (where it would be unrealistic to expect that there would be manual annotations of label candidates for that domain). We primarily use the in-domain results to gauge the relative quality of the cross-domain results.

We present the results in Table 2, displaying the performance of NETL and LGNB side by side for ease of comparison.[16] For each domain, the unsupervised baseline of NETL is based on the overlap of letter trigrams of the generated candidates with topic words (see Section 3.2). The unsupervised baseline of LGNB ranks the labels using a lexical association measure (Pearson's χ^2).

Looking at the performance of our method, we can see that the supervised system improves over the unsupervised baseline across all domains with the exception of a small drop observed in NEWS. Surprisingly, there is relatively little difference between the in-domain and cross-domain results for our method (but greater disparity for LGNB, especially over BOOKS; for NEWS, our cross-domain models actually outperform the in-domain model). The most consistent cross-domain results are generated when we combine all 3 domains, an unsurprising result given that it has access to the most training data, but encouraging in terms of having a single model which performs consistently across a range of domains.

We next compare NETL to LGNB, first focusing on the top-1 average rating metric. The most striking difference is the large improvement over PUBMED. LGNB attributed the poor performance over PUBMED to it being more domain-specific (and a poorer fit to Wikipedia) than the other domains, and suggested the need of biomedical experts for annotation. Our experiments found, however, that the performance of PUBMED is comparable to other domains. Additionally, the improvement in BOOKS is also quite substantial. Overall, NETL is more consistent across different domains and outperforms LGNB over 2 domains (BOOKS and PUBMED), and the difference between NETL and LGNB is small for NEWS and BLOGS. The other observation is the upper bound performance of NETL is uniformly better than that of LGNB, implying we are also generating better label candidates (we revisit this in detail in Section 5.2.1).

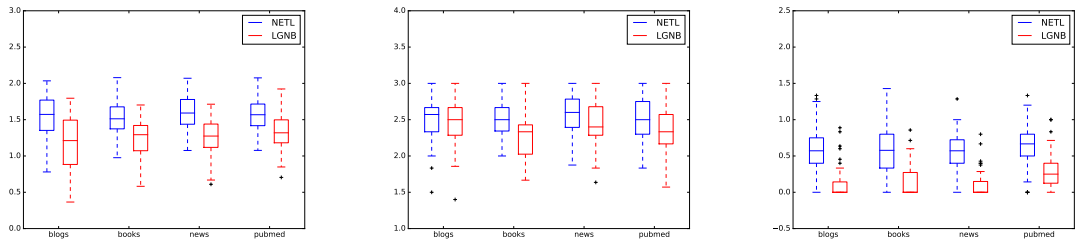
Moving to nDCG, the performance difference for nDCG-3/5 is largely indistinguishable for the two systems. LGNB, however, outperforms NETL in NEWS for nDCG-1 whereas NETL does better inPUBMED for nDCG-1 .

To give a sense of the sort of labels generated by NETL, we present a few topics and their top-ranked labels in Table 3.

5.2 Breaking Down NETL vs. LGNB

The results for NETL and LGNB in Table 2 conflate the candidate label selection and ranking steps, making it hard to get a sense of the relative impact of the different design choices implicit in the two sub-tasks. To provide a better comparison between the two methodologies, we present experiments

[16]LGNB results are taken directly from the original paper.



(a) Mean Ratings (b) Maximum Ratings (c) Minimum Ratings

Figure 1: Boxplots of ratings of candidates generated by NETL and LGNB.

evaluating the candidate generation and ranking method of the two systems separately.

5.2.1 Candidate Generation

In Table 2 we saw that NETL has a higher upper bound than that of LGNB, suggesting that the generated candidate labels were on average better. This is despite the average number of topic label candidates actually being higher for LGNB (25 vs. 19). Here, we present a more rigorous evaluation of the candidate generation method of both systems.

For each topic, we determine the mean, maximum and minimum label ratings for a given topic, and plot them in boxplots in Figure 1, aggregated per domain. The mean rating boxplot shows the average quality of candidates, while the maximum (minimum) rating boxplot reveals the average best (worst) quality of candidates that are generated by the two systems.

Looking at the boxplots, we see very clearly that NETL generates on average higher-quality candidates. Across all domains for mean, maximum and minimum ratings, the difference is substantial.

To provide a quantitative evaluation, we conduct one-sided paired t-tests to test the difference for all pairs in the boxplots. Except for the maximum ratings on BLOGS, all tests are significant ($p < 0.05$). These results demonstrate that NETL generates better candidates than LGNB (in all of the best-case, average-case and worst-case scenarios).

5.2.2 Candidate Ranking

Next, we directly compare the ranking method of NETL and LGNB. Using candidates generated by NETL, we re-rank the candidates using the ranking method of each of LGNB and NETL, and compare the results.

Both LGNB and NETL train an SVR re-ranker, using a partially-overlapping set of features. For LGNB, the ranking methodology uses 7 lexical association measures (PMI, Student's t-test, Dice's coefficient, Pearson's χ^2 test, log likelihood ratio, conditional and reverse conditional probability), 2 lexical features (the same 2 features that NETL uses: *NumWords* and *TopicOverlap*), and a search engine score based on Zettair. NETL, on the other hand, uses only 4 features: *LetterTrigram*, *PageRank*, *TopicOverlap* and *NumWords*.

For LGNB, we exclude the Zettair search engine score feature (as it was found to be an unimportant feature), and generate the lexical association features by parsing English Wikipedia. We train 2 SVR models using LGNB and NETL features. Results are presented in Table 4.

Using the same candidates, we see that NETL's features produce better rankings, outperforming LGNB's features across all domains. This shows that not only does NETL generate better candidates, but also ranks them better than LGNB.

6 Discussion

To better understand the contribution of each feature in NETL, we perform feature ablation tests (Table 5). An interesting observation is that different features appear to have different impact depending on the

Test Domain	Features	Top-1 Avg.	nDCG-1	nDCG-3	nDCG-5
BLOGS	LGNB	1.92	0.79	0.81	0.82
	NETL	**2.00**	**0.81**	**0.85**	**0.84**
BOOKS	LGNB	1.86	0.77	0.79	0.80
	NETL	**1.99**	**0.82**	**0.82**	**0.84**
NEWS	LGNB	1.87	0.75	0.79	0.81
	NETL	**2.02**	**0.80**	**0.84**	**0.85**
PUBMED	LGNB	1.89	0.77	0.79	0.81
	NETL	**1.99**	**0.81**	**0.81**	0.82

Table 4: Comparison of ranking performance with NETL features and LGNB features. Boldface indicates the better system between NETL and LGNB (with an absolute difference > 0.01).

Test Domain	BLOGS	BOOKS	NEWS	PUBMED
All Features	2.00	1.99	2.02	1.99
−*LetterTrigram*	1.99 (−.01)	1.96 (−.03)	2.02 (±.00)	1.93 (−.06)
−*PageRank*	1.93 (−.07)	1.980 (−.01)	2.00 (−.02)	2.00 (+.01)
−*TopicOverlap*	2.00 (±.00)	2.03 (+.04)	2.04 (+.02)	1.95 (−.04)
−*NumWords*	1.97 (−.03)	2.02 (+.03)	2.02 (±.00)	2.00 (+.01)

Table 5: Feature ablation results based on in-domain top-1 average ratings.

domain. Looking at BLOGS, we find that there is a considerable drop in top-1 average rating when we remove the *PageRank* feature. Similarly, ablating *LetterTrigram* appears to have a significant impact on PUBMED as well as some influence on BOOKS. As far as NEWS is concerned, we observe feature ablation does not play a big role. These observations indicate there is some degree of complementarity between these features, and that combining them produces robust and consistent performance across different domains.

Additionally, we explored using different numbers of topic terms when computing topic and title relevance for candidate ranking (we tested using top-5/10/15/20 topic terms). In general, we find that performance drops with the increase in topic terms. We also experiment with weighting each topic term with its word probability. We observed an improvement, although the difference is so marginal that we omit the results from the paper. Lastly, we tried computing relevance by first computing the centroid of topic terms before computing the cosine similarity with a candidate title. Again, we found little gain with this approach.

One feature type that we expect would have high utility is graph connectivity over the graphical structure of the Wikipedia categories or similar, along the lines of Hulpus et al. (2013). We leave this to future work. Methods based on keyphrase extraction such as Zhao et al. (2011) are also potentially worth exploring, although it remains to be seen whether notions such as "interestingness" benefit topic label selection.

7 Conclusion

We propose a neural embedding approach to automatically label topics using Wikipedia titles. Our methodology combines document and word embeddings to select the most relevant labels for topics. Compare to a state-of-the-art competitor system, our model is simpler, more efficient, and achieves better results across a range of domains.

References

Nikolaos Aletras and Mark Stevenson. 2013. Representing topics using images. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 158–167, Atlanta, USA.

Nikolaos Aletras, Timothy Baldwin, Jey Han Lau, and Mark Stevenson. 2014. Representing topics labels for exploring digital libraries. In *Proceedings of Digital Libraries 2014*, London, UK.

Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. 2008. On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM-08)*, pages 3–12, Washington, DC, USA.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2210–2216, Austin, USA.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison Wesley.

Karl Grieser, Timothy Baldwin, Fabian Bohnert, and Liz Sonenberg. 2011. Using ontological and document similarity to estimate museum exhibit relatedness. *ACM Journal of Computing and Cultural Heritage*, 3:10:1–10:20.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 363–371, Honolulu, USA.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using DBpedia. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 465–474, Rome, Italy.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4).

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226, Philadelphia, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 423–430, Sapporo, Japan.

Wanqiu Kou, Li Fang, and Timothy Baldwin. 2015. Automatic labelling of topic models using word vectors and letter trigram vectors. In *Proceedings of the 11th Asian Information Retrieval Societies Conference (AIRS 2015)*, pages 229–240, Brisbane, Australia.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems 25*, pages 2708–2716.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*, pages 1536–1545, Portland, USA.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, volume 14, pages 1188–1196, Beijing, China.

Qiaozhu Mei, Xuehua Shen, and Chengxiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007)*, pages 490–499, San Jose, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Newman, Timothy Baldwin, Lawrence Cavedon, Sarvnaz Karimi, David Martinez, and Justin Zobel. 2010a. Visualizing document collections and search results using topic mapping. *Journal of Web Semantics*, 8(2–3):169–175.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010b. Automatic evaluation of topic coherence. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010)*, pages 100–108, Los Angeles, USA.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank citation ranking: Bringing order to the web. Stanford Digital Libraries SIDL-WP-1999-0120.

Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Leah Findlater, and Kevin Seppi. 2016. ALTO: Active learning with topic overviews for speeding label induction and document labeling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1158–1169, Berlin, Germany.

Alison Smith, Tak Yeon Lee, Forough Poursabzi-Sangdeh, Leah Findlater, Jordan Boyd-Graber, and Niklas Elmqvist. to appear. Evaluating visual representations for topic understanding and their effects on manually generated labels. *Transactions of the Association for Computational Linguistics*.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 424–433, Philadelphia, USA.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li. 2011. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1*, pages 379–388, Portland, USA.