# Equipping Educational Applications with Domain Knowledge

**Tarek Sakakini*** **Hongyu Gong*** **Jong Yoon Lee***
**Robert Schloss†** **Jinjun Xiong†** **Suma Bhat***
*University of Illinois at Urbana-Champaign, USA
†T. J. Watson Research Center, IBM, USA
*{sakakini, hgong6, jlee642, spbhat2}@illinois.edu
†{rschloss, jinjun}@us.ibm.com

## Abstract

One of the challenges of building natural language processing (NLP) applications for education is finding a large domain-specific corpus for the subject of interest (e.g., history or science). To address this challenge, we propose a tool, Dexter, that extracts a subject-specific corpus from a heterogeneous corpus, such as Wikipedia, by relying on a small seed corpus and distributed document representations. We empirically show the impact of the generated corpus on language modeling, estimating word embeddings, and consequently, distractor generation, resulting in a better performance than while using a general domain corpus, a heuristically constructed domain-specific corpus, and a corpus generated by a popular system: BootCaT.

## 1 Introduction

Educational applications tend to target a specific subject, in other words, a specific domain, such as the medical domain in the case of (Jin et al., 2018). Thus, building these applications with underlying NLP algorithms, would typically require a large domain-specific corpus. Example uses of these large corpora are estimating language models (Rosenfeld, 2000), estimating word embeddings (Mikolov et al., 2013), and estimating document embeddings (Le and Mikolov, 2014). These estimations are central to several downstream applications including automatic speech recognition (Katz, 1987), machine translation (Koehn et al., 2003), and text categorization (Tang et al., 2015).

Previous findings, such as (McClosky, 2010), have shown that training NLP applications on a domain different from the target domain could prove detrimental to the performance of these applications. In order to help educational applications in specific disciplines such as science and history create a large, yet domain-specific corpus, we propose a domain extraction tool, Dexter[1], that extracts a domain-specific corpus from Wikipedia.

The algorithm, elaborated in Section 2, retrieves a set of documents from Wikipedia that are closest in discipline to a user-supplied small seed corpus. The size of this extracted set is a user-defined hyperparameter, and thus controls the trade-off between the specificity of the output corpus and its size. We empirically determine the favorable configuration of Dexter, demonstrate its benefits towards estimating word embeddings, and consequently distractor generation, as well as language models. We also show how, on the aforementioned tasks, Dexter outperforms BootCaT, a popular toolkit to automatically create an Internet-derived corpus (Baroni and Bernardini, 2004). Datasets used in this research are released for public use[2].

## 2 Method

Dexter's algorithm builds on the assumption that the distributed representation of two documents covering similar topics are closer in the vector space than two documents covering different topics. To test this assumption qualitatively, we embed all Wikipedia articles in $\mathbb{R}^{300}$ using Doc2Avg[3], and then map them to $\mathbb{R}^2$ using t-SNE (Maaten and Hinton, 2008) as shown in Figure 1. We see that the subset of science Wikipedia articles form a cluster, thus validating our assumption. Details on how science Wikipedia articles were identified are provided in Section 3.1. Building on this observation, Dexter takes a seed set of articles representing the target domain (e.g., science), then sorts Wikipedia articles in increasing order of distance to seed set, then returns the first $k$ documents as the extracted in-domain corpus.

[1]Publicly available at http://bit.ly/dexter-acl
[2]http://bit.ly/dexter-dataset-acl
[3]By averaging embeddings of words in document.



*Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 472–477
Florence, Italy, August 2, 2019. 

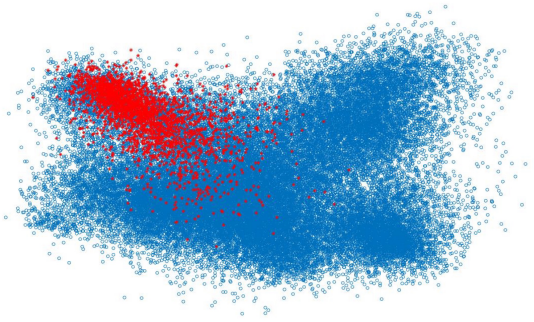


Figure 1: Clustering of science (red) Wikipedia articles among all (blue) Wikipedia articles in 2D.

Multiple design choices need to be considered carefully and are decided empirically in Section 3.2. These decisions are: The document representation method, the distance function to the seed set, and the seed corpus size. For the document representation, we consider: Doc2Vec (Le and Mikolov, 2014), Doc2Avg, Doc2wAvg[4], TFIDF (Leskovec et al., 2014), and LSA (Dumais, 2004). For the distance function, we consider: Mean, Min, Max, $90^{th}$ percentile, and $10^{th}$ percentile (the last two being robust to outliers). For the seed corpus size, we experiment with: $10$, $10^2$, and $10^3$.

## 3 Experiments

We perform three main experiments. The first is an intrinsic evaluation of Dexter, guiding our design choices. Second, we check the effect of the domain-specificity of the resulting word embeddings on the downstream educational task of distractor generation for science questions. Third, we evaluate the effect of Dexter on language modeling. Before we delve on the experimental details, we describe the process of labeling Wikipedia articles as science or not, and describe our competitive baseline: BootCaT.

### 3.1 Experimental Setup

**Dataset Preparation:** Wikipedia does not clearly partition its articles into different domains, but instead assigns a set of categories to each article. Wikipedia also organizes its categories as a directed graph, where, if category $b$ is a subcategory of category $a$, then there exists an edge $(a, b)$ (Schönhofen, 2009). Although it might seem natural to consider all descendants of a category (domain in a broad sense) to belong to that category, upon inspection we found the need for setting a depth limit. For example, "Stalking" is a third descendant of "Biology", although "Stalking" is not considered a "Biology"-related subject. Based on similar observations, we set the depth limit to two.

To identify science articles, we consider a list of science root categories, and all their subcategories up to a depth of 2. All articles labeled with any category in this list are considered science articles, amounting to 176,905 articles. This collected science corpus will be referred to as $C_D$, while the general Wikipedia corpus will be referred to as $C$.

We note that the corpus $C_D$ is created using heuristics on Wikipedia's taxonomy. In order to assess the extent to which the articles in $C_D$ belong to the discipline of science, we do the following. Two annotators assess the quality of $C_D$ by taking a subset of 1000 articles equally spread across depths: 0, 1, 2, and 3. Each article (depth anonymized) was given an integer score between 0 and 5 to reflect how much the content of the article is related to science. The inter-annotator agreement on the scores had a Pearson's correlation coefficient of 0.77 suggesting a reasonably high agreement on the scores. Upon comparing the average scores of articles at different depths, we found the score to be inversely proportional to the depth, with scores 4.35, 3.58, 3.21, and 2.4 for articles at depth 0, 1, 2, and 3 respectively. This further justifies our depth limit of 2, below which the average score suddenly drops below 3.

Also, for the purposes of our experiments, we split $C_D$ (176,905 articles) into three corpora: (1) $C_{seed}$ (1,000 articles having a depth of 0), used as the seed corpus for Dexter and the BootCaT baselines, (2) $C_{ho}$ (40,000 articles), which is a held-out dataset to be used for testing language models, and (3) $C_{silver}$[5] (136,905 articles) = $C_D - C_{ho}$, our training subset of $C_D$ to be used in language modeling experiments. $C_D$ reflects the quality of such a corpus heuristically-constructed using Wikipedia's taxonomy.

**Baseline:** One popular system used by researchers to extract a domain-specific corpus is BootCaT (Baroni and Bernardini, 2004), which operates on the World Wide Web. BootCaT generates queries to a search engine from user-supplied key phrases and parses the first $n$ pages retrieved for each query, where $n$ is set by the user.

Since Dexter requires seed articles instead of

[4]The embedding of a document is the average of its words' embeddings weighted by a word's TFIDF score

[5]The term silver is used, rather than gold, since we rely on heuristics rather than direct human supervision.

key phrases, we bridge the gap by a key phrase extraction algorithm (Rose et al., 2010) on the set of seed articles, by utilizing a publicly available implementation[6] with default parameters. We thus take $C_{seed}$ and extract the top-100 key phrases with less than 4 words, then feed them to BootCaT, leading to a domain-specific corpus (referred to as BootCat-KE). To avoid any possible noise introduced by the keyword extraction algorithm we consider another version of the BootCaT baseline but now with a manual set of 100 key phrases describing the domain of science. We take a list of science key phrases available online[7], and then randomly select 100 phrases. The corpus generated by this algorithm is referred to as BootCat-M.

### 3.2 Intrinsic Evaluation

Before we analyze Dexter's performance on downstream tasks and compare it to BootCaT, we study the intrinsic performance of Dexter under several design choices and conditions. Our evaluation of Dexter is based on the precision of the extracted articles averaged over 5 runs. Since we would be manipulating the seed set size, and to ensure Dexter's robustness under randomness, we artificially construct our seed set by taking a random subset of $C_D$ instead of using $C_{seed}$. That seed set is then used to algorithmically extract the rest of $C_D$ via Dexter. Accordingly, precision is calculated as the percentage of articles extracted, which belong to $C_D - C_{seed}$. As for recall, it is not measured since precision is sufficient as a comparison between methods assuming same number of documents extracted.

**Document representation**: We vary the document representation method while fixing the seed corpus size at $10^3$, and the distance function as Mean (c.f. Figure 2a). We observe that LSA and TFIDF are initially superior, but perform comparably to Doc2Avg and Doc2wAvg as $k$ increases. LSA is chosen due to its low-dimensionality, and superiority for modest $k$ values.

**Distance function**: We vary the distance function used while fixing the document representation to LSA and the seed corpus size to 1000 (c.f. Figure 2b). We observe that the $10^{th}$ percentile distance function leads to the best precision. We hypothesize that this is due to the $10^{th}$ percentile being robust to noise, and requiring closeness to only one subdomain of science rather than all at once.

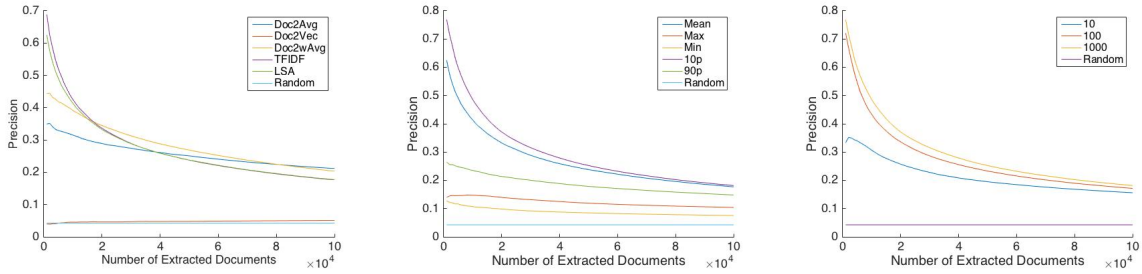
**Seed corpus size**: We vary the seed corpus size while fixing the document representation to LSA and the distance function as $10^{th}$ percentile (c.f. Figure 2c). We observe that a size of 100 was equally sufficient to 1000. This shows that Dexter does not require an unfeasibly large seed corpus size, which would have defeated the purpose.

### 3.3 Distractor generation

To better assess the impact of the extracted corpus by Dexter, we consider the task of distractor generation for multiple-choice questions (MCQs) (Stasaski and Hearst, 2017). Educators spend significant amount of time choosing suitable distractors for MCQs, where the distractors are the incorrect choices in an MCQ. Moreover, distractor generation is an essential task for automatic question generation. The choice of distractors is critical to the learning outcomes of students, since a misinformed selection of easy distractors could render questions non-challenging (Araki et al., 2016). The main aspect of the distractors' quality is their semantic similarity to the correct answer. The distractor cannot be a synonym of the answer and if too distant, it can be easily eliminated by the learner.

To automate this process, one considered methodology is relying on word embeddings to capture the semantics of the answer, and retrieve the distractors closest semantically to the answer (Araki et al., 2016). An essential component to the quality of these word embeddings is the domain of the training corpus, as any shift in domain would lead to a decrease in performance as noted in (Bollegala et al., 2015). To illustrate this further, we take the corpora $C$ (∼2.3B words) and $C_D$ (∼150M words), as described in Section 3.1, as well as the science corpus extracted by Dexter at size 150K (∼187M words) along with the corpora BootCaT-KE (∼986K words), and BootCaT-M (∼1.2M words). Also, to eliminate any effects of corpus size between Dexter and the BootCaT baselines, we downsample Dexter's corpus to the size of the BootCaT corpora ( 986K words). We then train six sets of word embeddings using FastText (Bojanowski et al., 2017), on each of the six aforementioned corpora. The quality of these word embeddings at capturing the semantics of science words is then measured on the task of distractor generation. Taking all questions from

[6] https://github.com/csurfer/rake-nltk
[7] http://sci2.esa.int/glossary/




(a) Effect of Document Representation (b) Effect of Distance Function (c) Effect of Seed Corpus Size

Figure 2: Effect of design choices on accuracy of extracted domain

| Corpus | Recall | Perplexity |
|---|---|---|
| $C$ | 17.43% | 431.78 |
| $C_{silver}$ | N/A | 334.57 |
| $C_D$ | 20.47% | N/A |
| BootCaT-KE | 15.28% | 3199.30 |
| BootCaT-M | 13.82% | 4586.80 |
| Dexter-Downsampled | 18.86% | 1117.34 |
| Dexter | 22.71% | 294.20 |

Table 1: Distractor recall@100 for word embeddings (middle) and perplexity of language models (right) trained on corpora of varying domain specificity.

three science questions datasets – 7,787 questions from ARC (Clark et al., 2018), 13,679 questions from SciQ (Johannes Welbl and Gardner, 2017), and 5059 questions from AI2-ScienceQuestions (Allen Institute for AI, 2017) – we check the recall of the distractors[8] in the top 100 most similar words to the answer of each question. Results are reported in Table 1.

As hypothesized, we notice that science-specific word embeddings trained on $C_D$ (20.47%) perform better than when trained on all of Wikipedia (17.43%). But it was surprising to observe that training on the science corpus generated by Dexter led to an even better performance (22.71%) than $C_D$ (20.47%). Since $C_D$ was heuristically constructed from human categorization, it might be the case that Dexter was able to capture the language characteristics of the science seed corpus better than heuristic methods operating on Wikipedia's taxonomy. The same two annotators mentioned in 3.1 manually labeled the extracted corpus (top 500 articles) using the same scale. The scores of the two annotators had a Perason's correlation coefficient of 0.66. Indeed the quality of the extracted corpus turned out higher than that of $C_D$, with an average score of 4.712. A less surprising result was Dexter's outperformance (22.71%) of both BootCaT baselines (15.28% and 13.82%) even when Dexter was downsampled (18.86%). This is expected as the automatic scraping of webpages by BootCaT introduces noisy artifacts into the corpus.

To qualitatively understand why word embeddings trained on domain-specific corpora outperform general ones we take a look at examples of polysemous words, and their word embeddings when trained on different corpora (c.f. Table 2). For example, the closest neighbors to the word "Force" when trained on $C$ are: "Forces", "Troops", and "Army", which reflect the military sense to the word "Force". When trained on $C_D$, the closest neighbors become: "Deflection", "Torque", and "Gravity", reflecting the scientific sense of the word "Force". Similarly, the closest neighbors to the word "Field", when trained on $C$, are: "Fields", "Football", and "Professional-sized", reflecting a sports field sense. Whereas, when trained on the extracted corpus by Dexter, neighbors of "Field" become: "Fields", "Magnetobiology", and "Ambipolar", reflecting a scientific sense of the word "Field".

### 3.4 Language modeling

Similar to the previous experiment, except for using $C_{silver}$ instead of $C_D$, we train 6 different trigram language models (Brown et al., 1992) on each corpus using kenlm (Heafield et al., 2013) and under default parameters[9]. We then test the perplexity of these language models on $C_{ho}$. The same pattern of comparative performances (c.f. Table 1) is noticed in language modeling, which reflects the impact of the quality of the domain-specific corpus on the variety of resources (language models and word embeddings) trained on

[8] We checked the recall of only one-word distractors since there is no straightforward method to retrieve phrases, as distractors, using distance over word embeddings. This does not affect our comparison study of different word embeddings.

[9] https://kheafield.com/code/kenlm/

| Word | Neighbors (General) | | | Neighbors (Science) | | |
|---|---|---|---|---|---|---|
| **Force** | Forces | Troops | Army | Deflection | Torque | Gravity |
| **Digest** | Review | Guide | Supplement | Digested | Extract | Metabolize |
| **Matter** | Matters | Subject | Debate | Particles | Materials | Universe |
| **Field** | Fields | Football | Professional-sized | Fields | Magnetobiology | Ambipolar |
| **Rock** | Punk | Pop | Indie | Rocks | Shoegazing | Screamo |
| **Cellular** | Cell | Signalling | Apoptosis | Cell | Organelle | Automata |

Table 2: Neighbors of polysemous scientific words when trained on the general Wikipedia (left), trained on $C_D$ (top right), and trained on the extracted science corpus by Dexter bottom right).

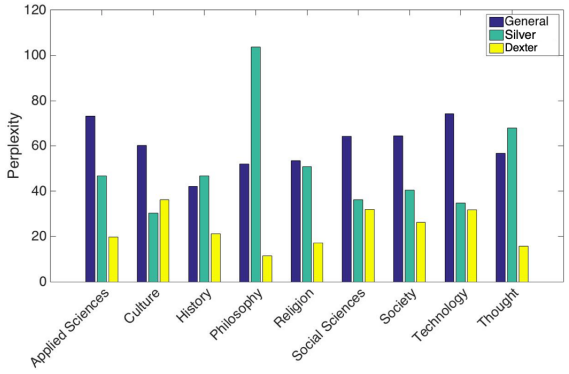



Figure 3: Perplexity scores for language models trained on General, Silver, and Dexter corpora across different domains.

it. We thus conclude that educational NLP applications in science can benefit from Dexter if their algorithm relies on a monolingual corpus. An example of an educational NLP application utilizing language models would be a machine translator for science webpages.

## 4 Generalization to other domains

To ensure Dexter's applicability to other educational domains, we repeated the experiment in Section 3.4 for educational domains other than science and show the results in Figure 3. We notice that the extracted corpus by Dexter is capable of training a significantly better language model than that trained on either the respective silver corpus or all of Wikipedia. The ineffectiveness of the general corpus is expected as the extracted corpus offers a more domain-specific training data. As for the more surprising outcome of inefficacy of the respective silver corpora, the reason seems to be the size of the silver corpus for most of these domains owing to the lack of articles in Wikipedia. The extracted corpus does not suffer from this limitation as its size is a hyperparameter set by the user, 100K in this case. With fewer than 100K in-domain articles, Dexter continues extracting articles that are close to but not in the domain leading to an extrapolated language model combining the benefits of the specificity of the small in-domain corpus and the generality of the large corpus.

## 5 Conclusion

Relying on off-the-shelf resources reduces the quality of educational NLP applications. To address this challenge, we offer to the community an aiding tool, Dexter, to extract a domain-specific corpus from Wikipedia. We show that our simple method outperforms in-domain corpora constructed heuristically using Wikipedia's taxonomy, or those constructed using popular systems scraping the World Wide Web.

## Acknowledgment

This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

## References

Allen Institute for AI. 2017. AI2 Science Questions v2.1. Accessed: 2019-04-16.

Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. 2016. Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1125–1136, Osaka, Japan. The COLING 2016 Organizing Committee.

Marco Baroni and Silvia Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In *LREC*, page 1313.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. *arXiv preprint arXiv:1505.07184*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Lifeng Jin, David King, Amad Hussein, Michael White, and Douglas Danforth. 2018. Using paraphrasing and memory-augmented models to combat data sparsity in question interpretation with a virtual patient dialogue system. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 13–23.

Nelson F. Liu Johannes Welbl and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Workshop on Noisy User-generated Text*.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

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

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

David McClosky. 2010. *Any domain parsing: automatic domain adaptation for natural language parsing*. Ph.D. thesis, Brown University.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Peter Schönhofen. 2009. Identifying document topics using the wikipedia category network. *Web Intelligence and Agent Systems: An International Journal*, 7(2):195–207.

Katherine Stasaski and Marti A. Hearst. 2017. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 303–312, Copenhagen, Denmark. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.