Software Framework for Topic Modelling with Large Corpora

Radim Řehůřek and Petr Sojka

Natural Language Processing Laboratory
Masaryk University, Faculty of Informatics
Botanická 68a, Brno, Czech Republic
{xrehurek,sojka}@fi.muni.cz

Abstract

Large corpora are ubiquitous in today's world and memory quickly becomes the limiting factor in practical applications of the Vector Space Model (VSM). In this paper, we identify a gap in existing implementations of many of the popular algorithms, which is their scalability and ease of use. We describe a Natural Language Processing software framework which is based on the idea of *document streaming*, i.e. processing corpora document after document, in a memory independent fashion. Within this framework, we implement several popular algorithms for topical inference, including Latent Semantic Analysis and Latent Dirichlet Allocation, in a way that makes them completely independent of the training corpus size. Particular emphasis is placed on straightforward and intuitive framework design, so that modifications and extensions of the methods and/or their application by interested practitioners are effortless. We demonstrate the usefulness of our approach on a real-world scenario of computing document similarities within an existing digital library DML-CZ.

1. Introduction

"Controlling complexity is the essence of computer programming."
Brian Kernighan (Kernighan and Plauger, 1976)

The *Vector Space Model (VSM)* is a proven and powerful paradigm in NLP, in which documents are represented as vectors in a high-dimensional space. The idea of representing text documents as vectors dates back to early 1970's to the SMART system (Salton et al., 1975). The original concept has since then been criticised, revised and improved on by a multitude of authors (Wong and Raghavan, 1984; Deerwester et al., 1990; Papadimitriou et al., 2000) and became a research field of its own. These efforts seek to exploit both explicit and implicit document structure to answer queries about document similarity and textual relatedness. Connected to this goal is the field of topical modelling (see e.g. (Steyvers and Griffiths, 2007) for a recent review of this field). The idea behind topical modelling is that texts in natural languages can be expressed in terms of a limited number of underlying *concepts* (or *topics*), a process which both improves efficiency (new representation takes up less space) and eliminates noise (transformation into topics can be viewed as noise reduction). A topical search for related documents is orthogonal to the more well-known "fulltext" search, which would match particular words, possibly combined through boolean operators.

Research on topical models has recently picked up pace, especially in the field of generative topic models such as Latent Dirichlet Allocation (Blei et al., 2003), their hierarchical extensions (Teh et al., 2006), topic quality assessment and visualisation (Chang et al., 2009; Blei and Lafferty, 2009). In fact, it is our observation that the research has rather gotten ahead of applications—the interested public is only just catching up with Latent Semantic Analysis, a method which is now more than 20 years old (Deerwester et al., 1990). We attribute reasons for this gap between research and practice partly to inherent mathematical complexity of the inference algorithms, partly to high computational demands of most methods and partly to the lack of a "sandbox" environment, which would enable practitioners to apply the methods to their particular problem on real data, in an easy and hassle-free manner. The research community has recognised these challenges and a lot of work has been done in the area of accessible NLP toolkits in the past couple of years; our contribution here is one such step in the direction of closing the gap between academia and ready-to-use software packages[1].

Existing Systems

The goal of this paper is somewhat orthogonal to much of the previous work in this area. As an example of another possible direction of applied research, we cite (Elsayed et al., 2008). While their work focuses on how to compute pair-wise *document similarities* from individual document representations in a scalable way, using Apache Hadoop and clusters of computers, our work here is concerned with how to scalably compute these document representations in the first place. Although both steps are necessary for a complete document similarity pipeline, the scope of this paper is limited to constructing topical representations, not answering similarity queries.

There exist several mature toolkits which deal with Vector Space Modelling. These include NLTK (Bird and Loper, 2004), Apache's UIMA and ClearTK (Ogren et al., 2008), Weka (Frank et al., 2005), OpenNLP (Baldridge et al., 2002), Mallet (McCallum, 2002), MDP (Zito et al., 2008), Nieme (Maes, 2009), Gate (Cunningham, 2002), Orange (Demšar et al., 2004) and many others.

These packages generally do a very good job at their intended purpose; however, from our point of view, they also suffer from one or more of the following shortcomings:

No topical modelling. Packages commonly offer supervised learning functionality (i.e. classification); topic inference is an unsupervised task.

[1] Interest in the field of document similarity can also be seen from the significant number of requests for a VSM software package which periodically crop up in various NLP mailing lists. Another indicator of interest are tutorials aimed at business applications; see web search results for "SEO myths and LSI" for an interesting treatment on Latent Semantic Indexing marketing.

Models do not scale. Package requires that the whole corpus be present in memory before the inference of topics takes place, usually in the form of a sparse term-document matrix.

Target domain not NLP/IR. The package was created with physics, neuroscience, image processing, etc. in mind. This is reflected in the choice of terminology as well as emphasis on different parts of the processing pipeline.

The Grand Unified Framework. The package covers a broad range of algorithms, approaches and use case scenarios, resulting in complex interfaces and dependencies. From the user's perspective, this is very desirable and convenient. From the developer's perspective, this is often a nightmare—tracking code logic requires major effort and interface modifications quickly cascade into a large set of changes.

In fact, we suspect that the last point is also the reason why there are so many packages in the first place. For a developer (as opposed to a user), the entry level learning curve is so steep that it is often simpler to "roll your own" package rather than delve into intricacies of an existing, proven one.

2. System Design

"Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."
Doug McIlroy (McIlroy et al., 1978)

Our choices in designing the proposed framework are a reflection of these perceived shortcomings. They can be explicitly summarised into:

Corpus size independence. We want the package to be able to detect topics based on corpora which are larger than the available RAM, in accordance with the current trends in NLP (see e.g. (Kilgarriff and Grefenstette, 2003)).

Intuitive API. We wish to minimise the number of method names and interfaces that need to be memorised in order to use the package. The terminology is NLP-centric.

Easy deployment. The package should work out-of-the-box on all major platforms, even without root privileges and without any system-wide installations.

Cover popular algorithms. We seek to provide novel, scalable implementations of algorithms such as TF-IDF, Latent Semantic Analysis, Random Projections or Latent Dirichlet Allocation.

We chose Python as the programming language, mainly because of its straightforward, compact syntax, multiplatform nature and ease of deployment. Python is also suitable for handling strings and boasts a fast, high quality library for numerical computing, *numpy*, which we use extensively.

Core interfaces

As mentioned earlier, the core concept of our framework is *document streaming*. A corpus is represented as a sequence of documents and at no point is there a need for the whole corpus to be stored in memory. This feature is not an afterthought on lazy evaluation, but rather a core requirement for our application and as such reflected in the package philosophy. To ensure transparent ease of use, we define *corpus* to be any iterable returning documents:

```
>>> for document in corpus:
>>>     pass
```

In turn, a document is a sparse vector representation of its constituent fields (such as terms or topics), again realised as a simple iterable:[2]

```
>>> for fieldId, fieldValue in document:
>>>     pass
```

This is a deceptively simple interface; while a corpus is allowed to be something as simple as

```
>>> corpus = [[(1, 0.8), (8, 0.6)]]
```

this streaming interface also subsumes loading/storing matrices from/to disk (e.g. in the Matrix Market (Boisvert et al., 1996) or SVMlight (Joachims, 1999) format), and allows for constructing more complex real-world IR scenarios, as we will show later. Note the lack of package-specific keywords, required method names, base class inheritance etc. This is in accordance with our main selling points: ease of use and data scalability.

Needless to say, both corpora and documents are not *restricted* to these interfaces; in addition to supporting iteration, they may (and usually do) contain additional methods and attributes, such as internal document ids, means of visualisation, document class tags and whatever else is needed for a particular application.

The second core interface are *transformations*. Where a corpus represents data, transformation represents the process of translating documents from one vector space into another (such as from a TF-IDF space into an LSA space). Realization in Python is through the dictionary [ ] mapping notation and is again quite intuitive:

```
>>> from gensim.models import LsiModel
>>> lsi = LsiModel(corpus, numTopics = 2)
>>> lsi[new_document]
[(0, 0.197), (1, -0.056)]

>>> from gensim.models import LdaModel
>>> lda = LdaModel(corpus, numTopics = 2)
>>> lda[new_document]
[(0, 1.0)]
```

[2] In terms of the underlying VSM, which is essentially a sparse field-document matrix, this interface effectively abstracts away from both the number of documents and the number of fields. We note, however, that the abstraction focus is on the number of *documents*, not fields. The number of terms and/or topics is usually carefully chosen, with unwanted token types removed via document frequency thresholds and stoplists. The hypothetical use case of introducing new fields in a streaming fashion doesn't come up as often in NLP.

2.1. Novel Implementations

While an intuitive interface is important for software adoption, it is of course rather trivial and useless in itself. We have therefore implemented some of the popular VSM methods, two of which we will describe here in greater detail.

Latent Semantic Analysis, LSA. Developed in late 80's in Bell Laboratories (Deerwester et al., 1990), this method gained popularity due to its solid theoretical background and efficient inference of topics. The method exploits co-occurrence between terms to project documents into a low-dimensional space. Inference is done using linear algebra routines for truncated Singular Value Decomposition (SVD) on the sparse term-document matrix, which is usually first weighted by some TF-IDF scheme. Once the SVD has been completed, it can be used to project new documents into the latent space, in a process called *folding-in*.

Since linear algebra routines have always been the front runner of numerical computing (see e.g. (Press et al., 1992)), some highly optimised packages for sparse SVD exist. For example, PROPACK and SVDPACK are both based on the Lanczos algorithm with smart reorthogonalizations, and both are written in FORTRAN (the latter also has a C-language port called SVDLIBC). Lightning fast as they are, adapting the FORTRAN code is rather tricky once we hit the memory limit for representing sparse matrices directly in memory. For this and other reasons, research has gradually turned to incremental algorithms for computing SVD, in which the matrix is presented sequentially—an approach equivalent to our *document streaming*. This problem reformulation is not trivial and only recently have there appeared practical algorithms for incremental SVD.

Within our framework, we have implemented Gorrell's Generalised Hebbian Algorithm (Gorrell, 2006), a stochastic method for incremental SVD. However, this algorithm proved much too slow in practice and we also found its internal parameters hard to tune, resulting in convergence issues. We have therefore also implemented Brand's algorithm for fast incremental SVD updates (Brand, 2006). This algorithm is much faster and contains no internal parameters to tune[3]. To our knowledge, our pure Python (numpy) implementation is the only publicly available implementation of LSA that does not require the term-document matrix to be stored in memory and is therefore independent of the corpus size[4]. Together with our straightforward document streaming interface, this in itself is a powerful addition to the set of publicly available NLP tools.

Latent Dirichlet Allocation, LDA. LDA is another topic modelling technique based on the bag-of-words paradigm and word-document counts (Blei et al., 2003). Unlike Latent Semantic Analysis, LDA is a fully generative model, where documents are assumed to have been generated according to a per-document topic distribution (with a Dirichlet prior) and per-topic word distribution. In practice, the goal is of course not generating random documents through these distributions, but rather inferring the distributions from observed documents. This can be accomplished by variational Bayes approximations (Blei et al., 2003) or by Gibbs sampling (Griffiths and Steyvers, 2004). Both of these approaches are incremental in their spirit, so that our implementation (again, in pure Python with numpy, and again the only of its kind that we know of) "only" had to abstract away from the original notations and implicit corpus-size allocations to be made truly memory independent. Once the distributions have been obtained, it is possible to assign topics to new, unseen documents, through our transformation interface.

[3]This algorithm actually comes from the field of image processing rather than NLP. Singular Value Decomposition, which is at the heart of LSA, is a universal data compression/noise reduction technique and has been successfully applied to many application domains.

[4]This includes completely ignoring the right singular vectors during SVD computations, as the left vectors together with singular values are enough to determine the latent space projection for new documents.

2.2. Deployment

The framework is heavily documented and can be accessed at `http://nlp.fi.muni.cz/projekty/gensim/`. This website contains sections which describe the framework and provide usage tutorials, as well as sections on download and installation instructions.

The framework is open sourced and distributed under an OSI-approved LGPL license.

3. Application of the Framework

"An idea that is developed and put into action is more important than an idea that exists only as an idea."
Hindu Prince Gautama Siddharta, the founder of Buddhism, 563–483 B.C.

3.1. Motivation

Many digital libraries today start to offer browsing features based on pairwise document content similarity. For collections having hundreds of thousands documents, computation of similarity scores is a challenge (Elsayed et al., 2008). We have faced this task during the project of The Digital Mathematics Library DML-CZ (Sojka, 2009). The emphasis was not on developing new IR methods for this task, although some modifications were obviously necessary—such as answering the question of what constitutes a "token", which differs between mathematics and the more common English ASCII texts.

With the collection's growth and a steady feed of new papers, lack of scalability appeared to be the main issue. This drove us to develop our new document similarity framework.

3.2. Data

As of today, the corpus contains over 61,293 fulltext documents for a total of about 270 million tokens. There are mathematical papers from the Czech Digital Mathematics Library DML-CZ `http://dml.cz` (22,991 papers), from the NUMDAM repository `http://numdam.org` (17,636 papers) and from the math part of arXiv `http://arxiv.org/archive/math` (20,666 papers). After filtering out word types that either appear less than five times in the corpus (mostly OCR errors) or in more than one half of the documents (stop words), we are left with 315,167 distinct word types. Although this is by no means an exceptionally big corpus, it already prohibits storing the sparse

term-document matrices in main memory, ruling out most available VSM software systems.

3.3. Results

We have tried several VSM approaches to representing documents as vectors: term weighting by TF-IDF, Latent Semantic Analysis, Random Projections and Latent Dirichlet Allocation. In all cases, we used the cosine measure to assess document similarity.

When evaluating data scalability, one of our two main design goals (together with ease of use), we note memory usage is now dominated by the transformation models themselves. These in turn depend on the vocabulary size and the number of topics (but not on the training corpus size). With 315,167 word types and 200 latent topics, both LSA and LDA models take up about 480 MB of RAM.

Although evaluation of the quality of the obtained similarities is not the subject of this paper, it is of course of utmost practical importance. Here we note that it is notoriously hard to evaluate the quality, as even the preferences of different types of similarity are subjective (match of main topic, or subdomain, or specific wording/plagiarism) and depends on the motivation of the reader. For this reason, we have decided to present all the computed similarities to our library users at once, see e.g. `http://dml.cz/handle/10338.dmlcz/100785/SimilarArticles`. At the present time, we are gathering feedback from mathematicians on these results and it is worth noting that the framework proposed in this paper makes such side-by-side comparison of methods straightforward and feasible.

4. Conclusion

We believe that our framework makes an important step in the direction of current trends in Natural Language Processing and fills a practical gap in existing software systems. We have argued that the common practice, where each novel topical algorithm gets implemented from scratch (often inventing, unfortunately, yet another I/O format for its data in the process) is undesirable. We have analysed the reasons for this practice and hypothesised that this partly due to the steep API learning curve of existing IR frameworks.

Our framework makes a conscious effort to make parsing, processing and transforming corpora into vector spaces as intuitive as possible. It is platform independent and requires no compilation or installations past Python+numpy. As an added bonus, the package provides ready implementations of some of the popular IR algorithms, such as Latent Semantic Analysis and Latent Dirichlet Allocation. These are novel, pure-Python implementations that make use of modern state-of-the-art iterative algorithms. This enables them to work over practically unlimited corpora, which no longer need to fit in RAM.

We believe this package is useful to topic modelling experts in implementing new algorithms as well as to the general NLP community, who is eager to try out these algorithms but who often finds the task of translating the original implementations (not to say the original articles!) to its needs quite daunting.

Future work will include comparison of the usefulness of different topical models to the users of our Digital Mathematical Library, as well as further improving the range, efficiency and scalability of popular topic modelling methods.

Acknowledgments

We acknowledge the support of grant MUNI/E/0084/2009 of the Rector of Masaryk University program for PhD students' research. We would also like to thank the anonymous reviewer for providing us with additional pointers and valuable comments.

5. References

J. Baldridge, T. Morton, and G. Bierner. 2002. The OpenNLP maximum entropy package. Technical report. `http://maxent.sourceforge.net/`.

Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit. *Proceedings of the ACL demonstration session*, pages 214–217.

David M. Blei and J. D. Lafferty. 2009. Visualizing Topics with Multi-Word Expressions. *Arxiv preprint `http://arxiv.org/abs/0907.1013`*.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022.

R. F. Boisvert, R. Pozo, and K.A. Remington. 1996. The matrix market formats: Initial design. Technical report, Applied and Computational Mathematics Division, NIST.

M. Brand. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. volume 31, Vancouver, British Columbia, CA.

H. Cunningham. 2002. GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254. `http://gate.ac.uk/`.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American society for Information science*, 41(6):391–407.

J. Demšar, B. Zupan, G. Leban, and T. Curk. 2004. Orange: From experimental machine learning to interactive data mining. *White Paper, Faculty of Computer and Information Science, University of Ljubljana*.

Tamer Elsayed, Jimmy Lin, and Douglas W. Oard. 2008. Pairwise Document Similarity in Large Collections with MapReduce. In *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pages 265–268, Morristown, NJ, USA. Association for Computational Linguistics.

E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. 2005. Weka: A machine learning workbench for data mining. *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, pages 1305–1314.

G. Gorrell. 2006. Generalized Hebbian algorithm for incremental Singular Value Decomposition in Natural Language Processing. In *Proceedings of 11th Conference of*

the European Chapter of the Association for Computational Linguistics (EACL), Trento, Italy, pages 97–104.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl 1):5228.

T. Joachims. 1999. SVMLight: Support Vector Machine. *SVM-Light Support Vector Machine* `http://svmlight.joachims.org/`, *University of Dortmund*.

Brian W. Kernighan and P. J. Plauger. 1976. *Software Tools*. Addison-Wesley Professional.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the Special Issue on the Web as Corpus. *Computational Linguistics*, 29(3):333–347.

Francis Maes. 2009. Nieme: Large-Scale Energy-Based Models. *The Journal of Machine Learning Research*, 10:743–746. `http://jmlr.csail.mit.edu/papers/volume10/maes09a/maes09a.pdf`.

A. K. McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`.

M. D. McIlroy, E. N. Pinson, and B. A. Tague. 1978. UNIX Time-Sharing System: Forward. *The Bell System Technical Journal*, 57(6 (part 2)), July/August.

P. V. Ogren, P. G. Wetzler, and S. J. Bethard. 2008. ClearTK: A UIMA toolkit for statistical natural language processing. *Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*, page 32.

C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. 2000. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 1992. *Numerical recipes in C*. Cambridge Univ. Press, Cambridge MA, USA.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):620.

Petr Sojka. 2009. An Experience with Building Digital Open Access Repository DML-CZ. In *Proceedings of CASLIN 2009, Institutional Online Repositories and Open Access, 16th International Seminar*, pages 74–78, Teplá Monastery, Czech Republic. University of West Bohemia, Pilsen, CZ.

M. Steyvers and T. Griffiths. 2007. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, pages 424–440.

Y. W. Teh, M. I. Jordan, M. J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

S. K. M. Wong and V. V. Raghavan. 1984. Vector space model of information retrieval: a reevaluation. In *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 167–185. British Computer Society, Swinton, UK.

T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. 2008. Modular toolkit for Data Processing (MDP): a Python data processing framework. *Frontiers in Neuroinformatics*, 2. `http://mdp-toolkit.sourceforge.net/`.