Software Framework for Topic Modelling

Radim Řehůřek and Petr Sojka

NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic

{xrehurek,sojka}@fi.muni.cz http://nlp.fi.muni.cz/projekty/gensim/

NLP Framework for VSM

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.

Introduction

"Controlling complexity is the essence of computer programming." Brian Kernighan [2]

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 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 their hierarchical extensions.

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 [4]

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. [3]).

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.

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:

```
>>> 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.

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)]
```

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, Latent Semantic Analysis, LSA and Latent Dirichlet Allocation, LDA.

The framework is heavily documented and is available from 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.

Application

"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.

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 [1]. We have faced this task during the project of The Digital Mathematics Library DML-CZ [5]. 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.

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.

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.

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.

References

[1] T. Elsayed, J. Lin, and D. W. Oard. 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, 2008. Association for Computational Linguistics.

[2] B. W. Kernighan and P. J. Plauger. *Software Tools*. Addison-Wesley Professional, 1976.

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

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

[5] P. Sojka. 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, 2009. University of West Bohemia, Pilsen, CZ.

[1] Support of grants MUNI/E/0084/2009 of MU Brno, 1ET200190513 of the Acad. of Sci. of the CR and MŠMT ČR LC536 is acknowledged.