PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

rsta.royalsocietypublishing.org

GATECloud.net: a platform for large-scale, open-source text processing on the cloud

Valentin Tablan, Ian Roberts, Hamish Cunningham and Kalina Bontcheva

Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, UK

Research

CrossMark
click for updates

Cite this article: Tablan V, Roberts I, Cunningham H, Bontcheva K. 2013 GATECloud.net: a platform for large-scale, open-source text processing on the cloud. Phil Trans R Soc A 371: 20120071.
http://dx.doi.org/10.1098/rsta.2012.0071

One contribution of 13 to a Theme Issue 'e-Science—towards the cloud: infrastructures, applications and research'.

Subject Areas:
artificial intelligence, e-Science, software

Keywords:
text mining, cloud computing, big data

Author for correspondence:
Valentin Tablan
e-mail: v.tablan@sheffield.ac.uk

Cloud computing is increasingly being regarded as a key enabler of the 'democratization of science', because on-demand, highly scalable cloud computing facilities enable researchers anywhere to carry out data-intensive experiments. In the context of natural language processing (NLP), algorithms tend to be complex, which makes their parallelization and deployment on cloud platforms a non-trivial task. This study presents a new, unique, cloud-based platform for large-scale NLP research—GATECloud.net. It enables researchers to carry out data-intensive NLP experiments by harnessing the vast, on-demand compute power of the Amazon cloud. Important infrastructural issues are dealt with by the platform, completely transparently for the researcher: load balancing, efficient data upload and storage, deployment on the virtual machines, security and fault tolerance. We also include a cost–benefit analysis and usage evaluation.

1. Introduction

The continued growth of unstructured content and the availability of ever more powerful computers have resulted in an increased need for researchers in diverse fields (e.g. humanities, social sciences, bioinformatics) to carry out language-processing and text-mining experiments on very large document collections (or *corpora*). An additional impetus is the availability of key datasets, e.g. Wikipedia and Freebase snapshots, which can help with experimental repeatability. Many of these datasets are impossible to process in reasonable time on standard computers such as desktop machines or individual servers.

In the context of natural language-processing (NLP) research, large-scale algorithms (also referred to as data-intensive or Web-scale NLP) are demonstrating increasingly superior results compared with approaches trained on smaller datasets, mostly thanks to addressing the data sparseness issue through collection of significantly larger numbers of naturally occurring linguistic examples [1]. The need for and the success of data-driven NLP methods to a large extent mirror recent trends in other research fields, leading to what is being referred to as 'the fourth paradigm of science' [2].

However, while researchers at big corporations (e.g. Google, Yahoo, Microsoft, IBM) have access to both Web-scale textual data (including Web query logs) and vast computing infrastructures, scientists from smaller research groups and universities are faced with major technological challenges when carrying out cutting-edge, data-driven, text-processing experiments. Cloud computing [3] is increasingly being regarded as a key enabler of the 'democratization of science' [2,4], giving researchers everywhere affordable access to computing infrastructures, which allow the deployment of significant compute power on an on-demand basis, and with no upfront costs.

However, NLP algorithms tend to be complex, which makes deployment on cloud platforms a specialized, non-trivial task, with its own associated costs in terms of significant time overhead and expertise required.

To answer these challenges, we have developed a novel, unique, cloud-based platform for large-scale NLP research—GATECloud.net. It aims to give researchers access to specialized software and enables them to carry out large-scale NLP experiments by harnessing the vast, on-demand compute power of the Amazon cloud. It also eliminates the need to implement specialized parallelizable text-processing algorithms. Important infrastructural issues are dealt with by the platform, completely transparently for the researcher: load balancing, efficient data upload and storage, deployment on the virtual machines, security and fault tolerance.

This study is structured as follows. Section 2 differentiates GATECloud.net from related work on data-intensive NLP and cloud computing. Section 3 motivates the need for an NLP platform-as-a-service (PaaS) and presents a number of requirements. Next, the architecture and implementation of GATECloud.net are discussed (§4) in relation to these requirements. The study concludes with a number of use cases and evaluation experiments (§5) and a discussion of future work (§6).

2. Large-scale text mining and compute clouds

Following the software-as-a-service (SaaS) paradigm from cloud computing [3], a number of text-processing services have been developed, e.g. OpenCalais,[1] Extractiv[2] and Alchemy API.[3] These mostly provide information extraction services, which are accessible programmatically (over a RESTful API) and charged based on the number of documents processed. However, they suffer from two key technical drawbacks. First, document-by-document processing over HTTP is inefficient on large datasets and is also limited to within-document text-processing algorithms. Second, the text-processing algorithms are pre-packaged: it is not possible for researchers to extend the functionality (e.g. adapt such a service to recognize new kinds of entities). Additionally, these text-processing SaaS sites come with daily rate limits, in terms of the number of API calls or documents that can be processed. Consequently, using these services is not just limited in terms of text-processing functionality offered, but also quickly becomes very expensive on large-scale datasets, especially Web pages and tweets that tend to be short but numerous.

At the same time, NLP researchers have started developing distributed algorithms for data-intensive text mining over terabyte-scale datasets. Many approaches adopt the MapReduce model for distributed computation, which can be deployed via Hadoop on clusters of affordable

[1]See http://www.opencalais.com.
[2]See http://extractiv.com.
[3]See http://www.alchemyapi.com.

compute servers. For instance, Lin [5] demonstrated using Hadoop to calculate word co-occurrence matrices; later Lin & Dyer [6] presented a number of distributed algorithms widely used in NLP tasks (Page-Rank, Expectation Maximization, hidden Markov models) and discuss their MapReduce implementations. Meng [7] demonstrated using Hadoop with NLTK [8]; word counts, name similarity and tf*idf are calculated and hidden Markov models are trained. However, having to adapt NLP algorithms in this way is very time-consuming. For example, van Gael and Bratieres from Cambridge University, UK, reported several person-months of effort to learn Hadoop and Amazon AWS and to rewrite their part-of-speech tagging algorithm [9]. An added complication is that not all NLP algorithms are actually implementable in the MapReduce paradigm, e.g. those requiring shared global state.

Apart from needing new algorithms, large-scale text processing also requires access to sufficiently large clusters of servers. Using on-demand compute power is now possible through Infrastructure-as-a-Service (IaaS) providers [10]. Arguably, the most successful commercial one is currently Amazon, which offers various kinds of virtual machine instances. However, making optimal use of virtual IaaS infrastructures for data-intensive NLP again comes with a significant overhead, e.g. having to learn the intricacies of Amazon's APIs for the Elastic Compute Cloud (EC2) and Simple Storage Services (S3).

PaaS [3] are a type of cloud computing service that insulates developers from the low-level issues of using IaaS effectively, while providing facilities for efficient development, testing and deployment of software over the internet, following the SaaS model. In the context of traditional NLP research and development, and pre-dating cloud computing by several years, similar needs were addressed through NLP infrastructures, such as GATE [11] and UIMA [12]. These infrastructures have accelerated significantly the pace of NLP research, by providing a number of reusable algorithms (e.g. rule-based pattern-matching engines, common machine-learning algorithms), free tools for low-level NLP tasks (e.g. tokenization, sentence identification) and in-built support for multiple input and output document formats (e.g. XML, PDF, DOC, RDF, JSON).

Some attempts have been made to extend UIMA functionality to use cloud computing; Zhou *et al*. [13] proposed a layered architecture built on UIMA and Hadoop, with the intention of allowing UIMA to operate over very large datasets. Thus far, only a proof of concept on topic detection has been described. Ramakrishnan *et al.* [14] explored the topic of scaling up processing for the SciKnowMine project. They discussed various possibilities, including cloud computing via the Behemoth project, which aims to make both UIMA and GATE functionality available on Hadoop. Luís & de Matos [15] aimed to make the benefits of cloud computing available outside established NLP frameworks. They described an approach to integrating various NLP tools and executing them in parallel. They commented that UIMA and GATE would benefit from adopting MapReduce. Laclavik *et al.* [16] demonstrated using Ontea [17] with Hadoop.

This study presents GATECloud.net—the adaptation of the GATE infrastructure to the cloud, following the PaaS paradigm. It enables researchers to run their NLP applications without the significant overheads of re-implementing their algorithms for MapReduce and understanding Amazon's IaaS APIs.

GATECloud.net is novel and unique in that it is currently the only open-source, cloud-based PaaS for large-scale text processing. Similar to the text-processing SaaS discussed earlier, it offers a growing number of pre-packaged NLP services. However, owing to it being a specialized NLP PaaS, GATECloud.net also supports a bring-your-own-pipeline option, which can be built easily by reusing pre-existing NLP components and adding some new ones. Moreover, GATECloud.net is the only cloud-based NLP platform that supports *the complete NLP development life cycle*. In addition to offering entity extraction services such as OpenCalais, our NLP PaaS also supports data preparation (e.g. HTML content extraction), manual corpus annotation, measurement of inter-annotator agreement, performance evaluation, data visualization, indexing and search of full text, annotations and ontological knowledge.

Crucially for researchers who run large-scale text-processing experiments only infrequently, there are no recurring monthly costs: instead, GATECloud.net is pay-per-use, billed per hour (owing to Amazon's per-hour charging model). There is also no daily limit on the number of documents to process or on document size. Consequently, processing costs depend on the *total data size*, not on the number of documents. This is particularly advantageous for bulk processing of Twitter data and other such numerous, but small-sized, texts. One downside of our per-hour billing approach is that it makes it harder for users to estimate their likely usage costs upfront. We return to this issue in §5, where we discuss how users can overcome this problem.

3. Towards an natural language-processing PaaS: requirements and methodology

NLP platforms, such as GATE and UIMA, have been hugely successful, thanks to the clean separation between low-level tasks such as data storage, data visualization, location and loading of components and execution of processes from the data structures and algorithms that actually process human language. They also reduce significantly the integration overheads by providing standard mechanisms for NLP components to be combined into complete pipelines and to communicate data about language, using open standards such as XML and RDF.

In a cloud computing context, developing an NLP PaaS requires careful consideration of the following additional requirements:

(i) *Straightforward deployment and sharing of NLP pipelines.* How can we achieve this transparently for the NLP developer, i.e. NLP applications developed on a desktop machine can run without any adaptation on the PaaS. In addition, developers need to be able to share easily their NLP pipelines as SaaS, with on-demand scalability and robustness ensured by the underlying NLP PaaS.

(ii) *Efficient upload, storage and sharing of large corpora.* An NLP PaaS needs to support a secure and efficient way for users to bulk upload, analyse and download large text corpora, i.e. batch processing over large datasets. In addition, users need to be able to share their large text corpora between different NLP pipelines, running on the PaaS—both for services bundled within the NLP PaaS and for services created by the developers themselves.

(iii) *Algorithm-agnostic parallelization.* How best to parallelize the execution of complex NLP pipelines, which could contain arbitrary algorithms, not all of which are implemented/suitable for MapReduce and Hadoop.

(iv) *Load balancing.* Determine the optimal number of virtual machines for running a given NLP application within the PaaS, given the size of the document collection to be processed and taking into account the considerable overhead of starting up new virtual machines on demand.

(v) *Security and fault tolerance.* As with any Web application, the NLP PaaS needs to ensure secure data exchange, processing and storage, as well as to be robust in the face of hardware failures and processing errors.

In addition to these technical requirements, an NLP PaaS needs to offer comprehensive methodological support to underpin the NLP application development life cycle:

(1) Create an initial prototype of the NLP pipeline, testing on a small document collection, using an NLP application development environment, running on a standard desktop or a local server machine.

(2) Crowd-source a gold-standard corpus for evaluation and/or training, using a Web-based collaborative corpus annotation tool, deployed as a service on the PaaS.

(3) Evaluate the performance of the automatic pipeline on the gold standard (either locally within the desktop development environment or through the manual annotation environment on the cloud). Return to step 1 for further development and evaluation cycles, as required.

(4) Upload the large datasets and deploy the NLP pipeline on the PaaS.

(5) Run the large-scale text-processing experiment and download the results as XML, JSON, RDF or schema.org formats. Optionally, an NLP PaaS could also offer scalable semantic indexing and search over the linguistic annotations and the document content.
(6) Lastly, analyse any errors and, if required, iterate again over the required system development stages, either on a local machine or on the NLP PaaS.

Next, we present GATECloud.net—a fully implemented NLP PaaS and discuss how specifically we chose to address the technical and methodological requirements discussed earlier.

4. GATECloud.net: an implemented natural language-processing PaaS

(a) An overview of the GATE natural language-processing infrastructure

The GATE framework [11] was chosen as the most suitable open-source NLP infrastructure to underpin the GATECloud.net PaaS, because it provides a unique combination of tools, addressing all the steps in the NLP application development methodology outlined earlier:

— a comprehensive and extensible NLP application development environment (GATE Developer), offering specialized user interfaces for visualization and editing of linguistic annotations, parse trees, ontologies and other NLP-specific resources (e.g. name lists, lexicons), as well as numerous tools for automating performance evaluation of language-processing components;
— numerous reusable text-processing components for many natural languages, underpinned by a finite state transduction language (JAPE) for rapid prototyping and efficient implementation of shallow NLP analysis methods, as well as an extensible machine-learning layer;
— a multi-paradigm repository (GATE Mímir) that can be used to index and search over text, linguistic annotations, semantic schemas (ontologies) and semantic meta-data. It supports queries that arbitrarily mix full-text, structural, linguistic and semantic constraints and scales up to terabytes of text through federated indexing; and
— a Web-based annotation environment (GATE Teamware) for collaborative creation of manually annotated corpora (needed for NLP algorithm training and for quantitative evaluation).

GATE is widely used for building text-processing applications in diverse domains, including bioinformatics [18], social media analysis [19] and humanities computing [20]. Recent projects have increasingly faced the problem of running GATE-based text processing on terabyte datasets [21]. At the same time, the multi-tier service-oriented architecture of GATE Teamware, coupled with its centralized workflow engine, has made its deployment and administration too complex and error prone for many researchers.

(b) The GATECloud.net architecture

The high-level approach taken in GATECloud.net[4] is to layer the GATE NLP infrastructure on top of Amazon's cloud infrastructure and services, much in the same way in which GATE insulates researchers from having to deal with data formats, storage and command-line processing, while also making it straightforward to create new and run text analysis algorithms.

The GATECloud.net architecture (figure 1) comprises the following main elements:

GATECloud.net Web application, a Website implemented using the Grails[5] Web application framework. Its main functions are providing security and user authentication, Web-based user interface for virtual servers management and annotation jobs management.

[4]Available at https://gatecloud.net.
[5]See http://grails.org.

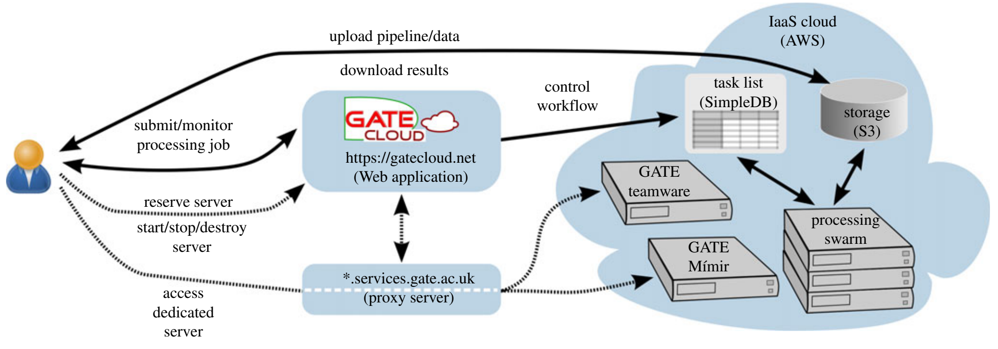


Figure 1. GATECloud.net architecture: dashed lines refer to supporting on-demand servers, whereas solid lines are for large-scale processing jobs. (Online version in colour.)

Amazon Web Services, a cloud infrastructure provider. EC2 provides virtual server instances that are used for hosting on-demand servers and for text-processing (annotation) jobs. *SimpleDB* is used for jobs and task management across distributed compute swarms. Finally, the S3 stores software images for our on-demand servers, datasets, processing results and processing reports in the case of annotation jobs.

services.gate.ac.uk, a specially configured Apache server that manages the `*.services.gate.ac.uk` subdomain. We use that to provide persistent DNS names for the on-demand servers belonging to the system users. This insulates users from the implementation details of the Amazon cloud, where each server instance gets a new IP address and a new DNS name whenever it is started. Amazon EC2 offers a service named *Elastic IPs* where a set of static IP addresses can be allocated to the virtual servers as needed. However, this is limited to five addresses as standard, while we needed to be able to support far larger numbers of machine reservations.

(c) On-demand servers

As discussed already, the GATE family of NLP tools includes server-side platforms (GATE Teamware and GATE Mímir), both having complex architectures and sets of dependencies that make them difficult to install and maintain. In many NLP projects, these tools are needed only for a limited time (e.g. Teamware is required only for manual annotation of training and evaluation corpora), which can make them uneconomical to install and maintain. A cloud-based SaaS deployment thus makes sense, as it reduces the costs in terms of administration effort and server hardware required.

Consequently, we created pre-installed Teamware and Mímir servers, running as Amazon EC2 instances, based on 64-bit Ubuntu Linux. Instead of virtual machine definitions (*Amazon machine images*, or *AMIs*) for server description, we chose to use *'recipes'* instead. Such a *recipe* comprises scripts and configuration data describing how to transform a standard base *AMI* into a dedicated Teamware or Mímir server. These scripts include steps such as downloading and installing the software and the appropriate support packages (e.g. Java or MySQL), initializing the databases and configuration files, attaching the persistent data storage associated with the user owning the machine. These steps are executed automatically each time a new server instance is started.

One downside of our approach is that it slightly increases the start-up time, typically by less than 5 min. However, the advantages are manifold:

Reduced administration costs: because our dedicated servers are built on top of standard AMIs, the responsibility for maintaining the basic operating system (e.g. installing security updates or upgrading software packages to newer versions) falls with the publisher of the AMI. We are currently using official Ubuntu Linux images, but any system that supports Java would work equally well.

Making a server reservation • The user requests a new reservation through the GATECloud.net web site, • a persistent data volume is created and associated with the user account, • a persistent server name is reserved for the user, • the user is notified in the web dashboard and through email.	*Destroying a server reservation* • The user requests the destruction of the reservation, • a check is made that the reservation is not currently associated with a running instance, • the previously created data volume is deleted, • the reserved server name removed from the database, • the user is notified.
Starting a server • The user requests the start-up of the server, • a new instance is started, • the previously created data volume is attached to the new instance, • the reserved server name is associated with the IP address of new instance, • the start-up process is confirmed to have completed successfully, • the user is notified.	*Stopping a server* • The user requests the stopping of the server, • the software services are stopped, • the persistent data partition is detached, • the server instance is terminated, • the user's account is updated with the associated costs, • the link between the reserved server name and the IP address of the terminated instance is removed, • the user is notified.

Figure 2. On-demand server workflows.

Flexibility: upgrading to a new version of the base operating system (or even starting to use a completely different OS) is simply a matter of changing the base AMI identifier used in our recipe.

Improved security: each server instance is essentially re-installed from scratch on each start-up. This reduces the scope for possible persistent attack vectors.

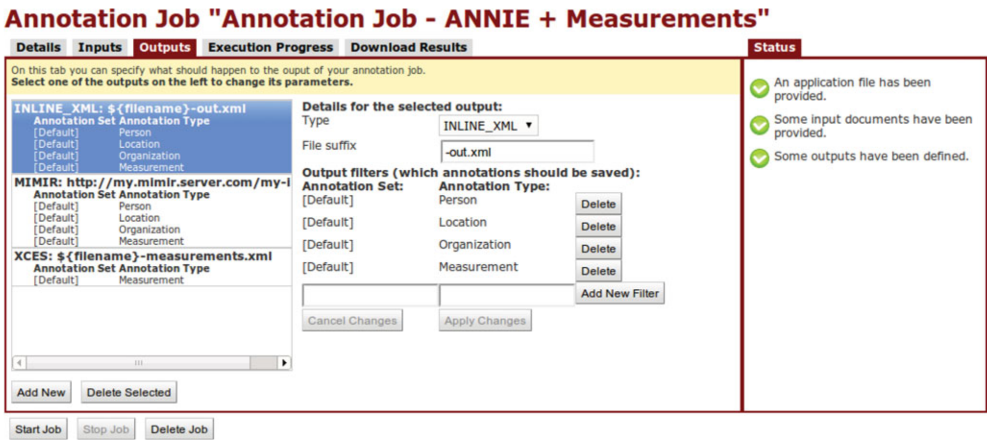
In order to provision a server, a GATECloud.net user needs to first make a reservation for it. Once reserved, the server appears in the user's dashboard, from where they can start it and stop it as required. While the server is running, the user incurs hourly charges. When the server is not required any more, the user can destroy the reservation, thus releasing all resources associated with it, such as the persistent data partition. The workflows used for these different operations are shown in figure 2.

(d) On-demand large-scale text processing

The other half of the GATECloud.net infrastructure is the support for processing of large document collections, using cloud computing. As discussed in §3, there are five major technical requirements that need to be met, coupled with many methodological ones, arising from the specifics of text processing.

The implementation of *Annotation Jobs* on GATECloud.net addressed most of these technical and methodological requirements, leaving researchers free to concentrate on their experiments. From a researcher's perspective, processing a document collection involves a few simple steps:

— upload the document collection (or point the system to the location where the data are available);



Annotation Job "Annotation Job - ANNIE + Measurements"

On this tab you can specify what should happen to the ouput of your annotation job.
Select one of the outputs on the left to change its parameters.

```
INLINE_XML: ${filename}-out.xml
    Annotation Set Annotation Type
    [Default]      Person
    [Default]      Location
    [Default]      Organization
    [Default]      Measurement
MIMIR: http://my.mimir.server.com/my-i
    Annotation Set Annotation Type
    [Default]      Person
    [Default]      Location
    [Default]      Organization
    [Default]      Measurement
XCES: ${filename}-measurements.xml
    Annotation Set Annotation Type
    [Default]      Measurement
```

Details for the selected output:

Type	INLINE_XML
File suffix	-out.xml

Output filters (which annotations should be saved):

Annotation Set:	Annotation Type:	
[Default]	Person	Delete
[Default]	Location	Delete
[Default]	Organization	Delete
[Default]	Measurement	Delete

✓ An application file has been provided.
✓ Some input documents have been provided.
✓ Some outputs have been defined.

Figure 3. Web-based job editor. (Online version in colour.)

— upload a GATE-based processing pipeline to be used (or choose one provided as SaaS on GATECloud.net); and
— press the '*Start*' button.

While the job is running, a regularly updated execution log is made available in the user's dashboard. Upon job completion, an email notification is also sent. Most of the implementation details are hidden away from the user, who interacts with the system through a Web-based job editor, depicted in figure 3.

Next, we discuss how GATECloud.net PaaS meets the requirements from §3:

(i) *Pipeline deployment and sharing:* GATECloud.net can run any NLP pipeline developed on the researcher's desktop, after it has been packaged by the GATE Developer environment [11], using the application export option 'Export for GATECloud.net'. This is an automated process that builds a self-contained zip file, including all text-processing modules and the linguistic data and ontologies required by them. In other words, there is no additional implementation effort required, in order to deploy a GATE-based NLP pipeline on the cloud-based NLP PaaS and execute it on a large-scale dataset.

(ii) *Efficient data management:* from an implementation perspective, job execution requires access to potentially large amounts of data, including the text-processing application file(s), the document collection to be processed, the execution reports and the results files (if any are produced). We chose to store all these data in a separate S3 '*location*' that is created especially for each job and cannot be accessed by other jobs. Once the job completes, users get a grace period (by default 10 days, but this can be changed) during which they can download the results. When this time has elapsed, the job's S3 location is deleted entirely in order to save storage costs. Alternatively, the user can choose to provide their own S3 location for storing the job data (with the appropriate permissions for the GATECloud.net AWS user account); in that case, no automatic deletion takes place.

(iii) *Parallelization* and (iv) *Load balancing:* job execution is supported by multiple interconnected, parallel workflows, which are described next.

Job management: each *Job* advances through a series of states during its life cycle. The jobs management process is implemented by the GATECloud.net Web application and deals with taking the appropriate steps to transition each active job from one state to the next. Each step usually involves generating new *Tasks*, and queuing them for execution. A *Task* is a piece of work that needs to be performed, and that is executed by a *Node* (a server instance in the cloud). Once all the *Tasks* belonging to a *Job* in a given state have completed, the *Job* can move to the next state.

Swarm load management: Swarms are sets of cloud instances (*Nodes*) that have identical configurations and that execute *Tasks* from the same task queue. Each *Swarm* has a target *load factor*, defined as (active tasks + pending tasks)/active nodes. A *load factor* of 1 indicates the intention of allocating a separate node for each running task, whereas a value of 2 means that the system would aim to have, for example, five running nodes for a list of 10 tasks. The *Swarm* management process (also part of the GATECloud.net Web application) starts new *Node* instances associated with all of the registered *Swarms* as required, in order to keep the actual *load factors* as close as possible to the target values.

Node workflow: each *Node* instance is configured during start-up with the tasks queue it should use. Once started, its workflow is a simple loop collecting the first pending *Task* from the queue, and executing it. When no more pending tasks have been available for a while, the node shuts itself down automatically. Each *Task* has an associated state and a time stamp. When a *Node* picks up a task for execution, it changes its state from `pending` to `active`. It also updates the time stamp at regular intervals while the *Task* is being executed. If a *Node* crashes for whatever reason, any *Task* it was working on will be left in an `active` state; however, its time stamp will stop being updated. Active *Tasks* with an old time stamp are presumed to have failed and will be rescheduled for execution three times. If a *Task* keeps failing to execute, then its state is changed to `failed` and it is not scheduled again, thus avoiding infinite loops.

The *Tasks* queue is implemented as an *Amazon SimpleDB* domain accessible by both the GATECloud.net Web application and all processing *Nodes*. Concurrency control is implemented by associating a `version` attribute with each item in the domain. The `version` starts at 0 and is incremented with each write operation. Completing the picture are the atomic conditional update operations provided by *SimpleDB*, which are used to make sure new values are written only if the `version` has not been independently changed since the old values were read.

(v) *Security:* because the PaaS allows researchers to upload their own NLP pipelines, we have no control over the software included within: it could contain poorly written, insecure or even malicious code. Consequently, security is a major concern and we are addressing this by judicious privilege isolation. The system account, which has access to cloud login credentials and system settings, is only executing code that has been produced by the platform authors and security audited. User data and user processes are executed by a separate OS-level user, which has very restricted permissions, allowing file access only within a given data directory. All the data and processes owned by this user are destroyed upon completion of each *Task*. This ensures that data and software belonging to different researchers are never present at the same time on the same *Node* machine instance.

While stored on the cloud, the user data are protected by the security procedures instituted by the cloud infrastructure provider. All the transfers between the cloud storage, the processing nodes and the user's computer are done via an encrypted channel, using SSL.

5. Use cases and experiments

In order to quantify the performance gains from running text-processing algorithms on large datasets on GATECloud.net, we carried out experiments on three document collections: 50 million tweets (very short texts), 20 000 news articles (medium-sized texts) and 100 000 patents (larger sizes, up to 6 MB). With respect to data sizes, the statistics for the three datasets are as follows:

Twitter We randomly selected 50 million tweets from a 1 TiB dataset. The Twitter feed was first simplified by preserving only the tweet text, tweet ID and the ID of the author, while discarding all other metadata fields. After conversion to XML, the size of the resulting corpus was 6 GiB. The 50 million tweets were chosen to be a usefully sized dataset, 10 times larger than the one used by Abel *et al.* [22] and 10 times smaller than the one used by Laniado & Mika [23]. Additional experiments, not detailed here owing to space constraints, showed that processing time scales linearly with the number of tweets, on all hardware configurations (see below). Consequently, the performance and cost figures reported here can be used to easily estimate the corresponding values for smaller or larger tweet datasets.

News The news corpus comprised 20 000 HTML pages (1.31 GiB), collected from the websites of news broadcasters BBC and CNN, and UK newspapers *The Independent* and *The Guardian*. The shortest document was just nine characters (from the CNN website), whereas the longest was 230 KiB. Most news articles are well clustered around the middle, with an average size of 68.7 KiB.

Patents The corpus contained 100 000 patent documents, with an overall size of 5.47 GiB. The mean document size is around 58 KiB, with the smallest document being just under 3 KiB (containing only the abstract) and the largest at 5.94 MiB. The majority of the patent documents are clustered around the middle, with the lower quartile of 22 KiB, median of 43 KiB and the upper quartile of 70 KiB.

The news and Twitter datasets were annotated for named entities with the standard *ANNIE* entity annotation pipeline [11], deployed as SaaS within GATECloud.net. For the patents dataset, we reused a pre-existing text-processing pipeline [21] that recognizes patent-specific types, including references to other patents, scientific publications, measurement expressions, patent sections, claims, examples, references to figures and tables.

In order to carry out a cost–benefit evaluation of GATECloud.net, we ran three sets of experiments, where each dataset was processed on the following hardware configuration:

Desktop *Lenovo ThinkCentre M58p* desktop computer, Intel Xeon E5502 1.86 GHz CPU (two cores), 4 GiB RAM, 320 GiB HDD Serial ATA II, cost ≈£1300. This is currently a standard desktop configuration for researchers in our laboratory.

Server *HP ProLiant DL385 G7* server, AMD Opteron 2.3 GHz CPU (12 cores), 32 GiB RAM, 2 TiB disk space, cost ≈£4800. For these experiments, we had access to six of the processing cores only.

Cloud *GATECloud.net* swarm, using 10 *Extra Large* instances on the Amazon cloud. Each node has 15 GiB of RAM, 8 EC2 Compute Units (four virtual cores with two EC2 Compute Units each), 1690 GiB of local instance storage, 64-bit platform. One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. For running costs, see table 1.

The only exception is the patents dataset that was too large to be processed on the desktop computer in a reasonable amount of time.

Table 1 reports the results of these three sets of experiments along several dimensions. *CPU time* measures the total amount of time spent, by any thread, processing documents, whereas *computer time* is the time taken by a given machine. On machines with multiple cores, computer time will be lower than the corresponding CPU time, as multiple parallel execution threads are used. *Clock time* represents the time interval between starting the process and its completion, as measured by an external clock. This differs significantly from the other measurements only in the case of the cloud-based experiments, where processing was distributed across several machines. On the desktop machine, even though it has two cores, only one of them was used as a result of running the experiments as standard batch processes, using a single execution thread. Consequently, the desktop running times are always the same for CPU time, computer time and clock time. The *clock time* column shows the overall time saving owing to using more powerful hardware configurations: from a desktop computer to a server, then to the cloud.

In terms of cost, the price per gigabyte for named entity recognition is very low (between £1.35 and £1.50). The patents application is significantly more complex and, therefore, its cost per gigabyte processed is approximately double. Similarly, its performance figures should not be compared directly with those obtained on news and tweets. The more general point is that the cost per gigabyte of text processed on GATECloud.net varies widely depending on the complexity

Table 1. Experiment results.

		time (hh:mm:ss)			speed	cost (£)	
		CPU time	compute time	clock time	(KiB/s)	£/GiB	total
experiment 1: patents	desktop	n.a.	n.a.	n.a.			
100 000 patent documents	server	91:42:00	18:39:54	18:39:54	85.33		
	cloud	162:38:00	16:56:37	02:03:32	773.6	£3.08	£16.83
experiment 2: news	desktop	05:20:19	05:20:19	05:20:19	71.52		
20 000 patent documents	server	04:43:00	03:08:00	03:08:00	121.86		
	cloud	07:47:00	01:21:20	00:35:31	645.04	£1.51	£1.98
experiment 3: tweets	desktop	32:28:46	32:28:46	32:28:46	52.80		
50 000 000 patent documents	server	22:16:15	03:19:12	03:19:12	516.53		
	cloud	40:08:00	07:00:14	01:25:46	1199.69	£1.35	£7.92

of the underlying text-processing algorithms. In addition, specifically on the patents dataset, a significant amount of time was spent on processing the few larger documents of over 1 MB in size. Time complexity for co-reference resolution is higher than linear; so it tends to be much more time consuming on such large documents owing to the large number of candidate entities that need to be checked. The time complexity of most other algorithms tends to be near linear with respect to document size.

It is, of course, required that users are able to estimate the cost of processing a given document collection. Given that each document is processed independently from the rest, the time taken to process a collection can be approximated by multiplying the total number of documents and the average execution time for a document.[6] The average execution time can be estimated by processing a small but statistically representative collection sample. Care should be taken when selecting the sample as, depending on what the actual processing consists of, document length may not be a good indicator of time complexity.

Compared with cloud infrastructure providers, GATECloud.net is simplifying cost estimations by hiding details such as the cost of data storage and data transfers behind a simple cost model based exclusively on CPU-hours.

When looking at CPU time used, this always tends to be higher on the cloud than the CPU times for the desktop and server. This is due to the overhead in using distributed computing, specifically mainly due to the need to split the large datasets into batches that run on separate nodes, as well as efficiency lost to virtualization. In addition, the actual hardware specifications of the Amazon *Extra Large* virtual machines are not as good as those of the server, but are quite comparable to our desktop configuration.

As can be seen from the time statistics, the major benefit of using GATECloud.net comes from the significant reduction in *Clock Time* taken for each of the experiments. The speed-up compared with the desktop configuration is between 10- and 20-fold. For example, the processing time for the news dataset is reduced down from over 5 h to 35 min, which can help significantly not only for processing large-scale datasets but also during the development of the algorithms by reducing the time taken by the develop–evaluate cycles. The time reduction is even greater on the tweets dataset, where time goes down from 32 h to 1 h and 25 min.

With respect to the gains made by using GATECloud.net instead of a local powerful server, there are also significant benefits (fivefold speed-up on the news set and 10-fold on the patents data). The benefits are less pronounced on tweets, owing to their large number and small

[6]The actual formula is more complex, but this is the dominant term.

size. In general, GATECloud.net has been optimized for processing medium- to large-sized documents, where the benefits are most pronounced. In future work, we will be working towards improving the infrastructure's performance on large collections of smaller documents.

An independent GATECloud.net benchmarking experiment was carried out by a team of researchers from the UK Food and Environment Research Agency.[7] They started by building a specialized text annotation pipeline, using GATE Developer. This included over 20 different rule-based components, as well as some of the low-level linguistic processing offered by GATE's standard tools. The document collection consisted of 261 260 documents ranging from 1 KiB to 2.5 MiB of text. Of all the documents, 14 (i.e. 0.005%) failed to complete successfully, owing to various text-processing exceptions. Their *CPU time* was just over 13 h (786 min), whereas the *clock time* value was 1 h and 20 min, again a 10-fold speed-up.

6. Conclusions and future work

This study motivated the need for a specialized NLP PaaS, identified a set of requirements, and presented our implementation of such an NLP PaaS – GATECloud.net. This is aimed at helping researchers to carry out data-intensive text-processing experiments on the cloud.

The platform has been made available to the public as a beta service. During its first six months of operation (between mid-June and mid-December 2011), there were 114 registered users. The number of processed documents was 4.7 million,[8] part of 302 annotation job runs. The accumulated server time used for both annotation jobs and dedicated servers was 430 h. This level of usage indicates a need for such tools and a clear interest from researchers and the wider community.

Currently, we are working on adding a programming API to GATECloud.net, so that data upload, processing and download can all be done automatically, outside the Web interface. This will allow tighter integration with legacy workflows and higher levels of automation. Other future work will focus on going beyond batch-oriented processing of documents, towards handling streaming data. Integration of MapReduce-based NLP algorithms is also being considered.

This work has been partially supported by an EPSRC/JISC grant (no. EP/I034092/1) funding pilot projects in cloud computing. K.B. is supported by a Career Acceleration Fellowship from the Engineering and Physical Sciences Research Council (no. EP/I004327/1).

References

1. Halevy A, Norvig P, Pereira F. 2009 The unreasonable effectiveness of data. *IEEE Intell. Syst.* **24**, 8–12. (doi:10.1109/MIS.2009.36)
2. Barga R, Gannon D, Reed D. 2011 The client and the cloud: democratizing research computing. *IEEE Internet Comput.* **15**, 72–75. (doi:10.1109/MIC.2011.20)
3. Dikaiakos MD, Katsaros D, Mehra P, Pallis G, Vakali A. 2009 Cloud computing: distributed internet computing for IT and scientific research. *IEEE Internet Comput.* **13**, 10–13. (doi:10.1109/MIC.2009.103)
4. Foster I. 2011 Globus online: accelerating and democratizing science through cloud-based services. *IEEE Internet Comput.* **15**, 70–73. (doi:10.1109/MIC.2011.64)
5. Lin J. 2008 Scalable language processing algorithms for the masses: a case study in computing word co-occurrence matrices with MapReduce. In *Proc. Conf. on Empirical Methods in Natural Language Processing, EMNLP'08, Honolulu, HI, 25–27 October 2008*, pp. 419–428. Stroudsburg, PA: Association for Computational Linguistics.
6. Lin J, Dyer C. 2010 *Data-intensive text processing with MapReduce*. Synthesis Lectures on Human Language Technologies. San Francisco, CA: Morgan & Claypool.
7. Meng X. Tutorial on MapReduce framework in NLTK. Technical report. See http://nltk.googlecode.com/svn/trunk/nltk_contrib/nltk_contrib/hadoop/doc/.

[7]See http://www.fera.defra.gov.uk/.

[8]For the tweets experiments, we created input documents from groups of 1000 tweets. Hence, the 50 000 000 tweets only count as 50 000 documents.

8. Loper E, Bird S. 2002 Nltk: the natural language toolkit. *CoRR* cs.CL/0205028. http://arxiv.org/abs/cs.CL/0205028.
9. Hammond M, Hawtin R, Gillam L, Oppenheim C. 2010 Cloud computing for research. Final report. Technical report, JISC. See http://www.jisc.ac.uk/whatwedo/programmes/researchinfrastructure/usingcloudcomp.aspx.
10. Li A. 2011 Comparing public cloud providers. *IEEE Internet Comput.* **15**, 50–53. (doi:10.1109/MIC.2011.36)
11. Cunningham H. *et al.* 2011 *Text processing with GATE* (*v. 6*). See http://gate.ac.uk/ books.html.
12. Ferrucci D, Lally A. 2004 UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.* **10**, 327–348. (doi:10.1017/S1351324904003523)
13. Zhou B, Jia Y, Liu C, Zhang X. 2010 A distributed text mining system for online web textual data analysis. In *Int. Conf. Cyber-enabled Distributed Computing and Knowledge Discovery (CyberC), Huangshan, China, 10–12 October 2010*, pp. 1–4. Los Alamitos, CA: IEEE Computer Society.
14. Ramakrishnan C *et al.* 2010 Building the scientific knowledge mine (SciKnowMine): a community-driven framework for text mining tools in direct service to biocuration. In *Proc. New Challenges for NLP Frameworks, LREC 2010, Valletta, Malta, 22 May 2010*, pp. 9–14. Valetta, Malta: ELRA.
15. Luís T, de Matos DM. 2009 High-performance high-volume layered corpora annotation. In *Proc. 3rd Linguistic Annotation Workshop, ACL-IJCNLP '09, Singapore, 2–7 August 2009*, pp. 99–107. Stroudsburg, PA: Association for Computational Linguistics.
16. Laclavik M, Seleng M, Hluchy L. 2008 Towards large scale semantic annotation built on MapReduce architecture. In *Computational science—ICCS 2008* (eds M Bubak, G van Albada, J Dongarra, P Sloot), pp. 331–338. Lecture Notes in Computer Science, vol. 5103. Berlin, Germany: Springer.
17. Laclavik M, Seleng M, Gatial E, Balogh Z, Hluchy L. 2007 Ontology based text annotation: OnTeA. In *Information modelling and knowledge bases XVIII* (eds M Duží, H Jaakkola, Y Kiyoki, H Kangassalo), pp. 311–315. Frontiers in Artificial Intelligence and Applications, vol. 154. Amsterdam, The Netherlands: IOS Press.
18. Johansson M *et al.* 2009 Using prior information attained from the literature to improve ranking in genome-wide association studies. In *Proc. 59th Annual Meeting of the American Society of Human Genetics, Honolulu, HI, 20–24 October 2009*.
19. Saggion H, Funk A. 2009 Extracting opinions and facts for business intelligence. *RNTI J.* E(**17**) 119–146.
20. Risse T, Dietze S, Maynard D, Tahmasebi N. 2011 Using events for content appraisal and selection in web archives. In *Proc. DeRiVE 2011: Workshop in conjunction with the 10th Int. Semantic Web Conference 2011, 23 October 2011, Bonn, Germany*. See http://ceur-ws.org/Vol-779/derive2011_submission_5.pdf.
21. Agatonovic M, Aswani N, Bontcheva K, Cunningham H, Heitz T, Li Y, Roberts I, Tablan V. 2008 Large-scale, parallel automatic patent annotation. In *Proc. 1st Int. CIKM Workshop on Patent Information Retrieval— PaIR '08, Napa Valley, CA, 30 October 2008*.
22. Abel F, Gao Q, Houben G-J, Tao K. 2011 Semantic enrichment of twitter posts for user profile construction on the social web. *ESWC* **2**, 375–389.
23. Laniado D, Mika P. 2010 Making sense of Twitter. In *The Semantic Web: ISWC 2010* (eds P. Patel-Schneider, Y Pan, P Hitzler, P Mika, L Zhang, J Pan, I Horrocks, B Glimm), pp. 470–485. Lecture Notes in Computer Science, vol. 6496. Berlin, Germany: Springer-Verlag.