Knowledge-Based Systems 79 (2015) 36–42

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Big data for Natural Language Processing: A streaming approach

Rodrigo Agerri *, Xabier Artola, Zuhaitz Beloki, German Rigau, Aitor Soroa

IXA NLP Group, University of the Basque Country (UPV/EHU), Donostia-San Sebastián, Spain

ARTICLE INFO

Article history:
Received 30 March 2014
Received in revised form 22 October 2014
Accepted 8 November 2014
Available online 20 November 2014

Keywords:
Natural Language Processing
Distributed NLP architectures
Big data
Storm
NLP tools

ABSTRACT

Requirements in computational power have grown dramatically in recent years. This is also the case in many language processing tasks, due to the overwhelming and ever increasing amount of textual information that must be processed in a reasonable time frame. This scenario has led to a paradigm shift in the computing architectures and large-scale data processing strategies used in the Natural Language Processing field. In this paper we present a new distributed architecture and technology for scaling up text analysis running a complete chain of linguistic processors on several virtual machines. Furthermore, we also describe a series of experiments carried out with the goal of analyzing the scaling capabilities of the language processing pipeline used in this setting. We explore the use of Storm in a new approach for scalable distributed language processing across multiple machines and evaluate its effectiveness and efficiency when processing documents on a medium and large scale. The experiments have shown that there is a big room for improvement regarding language processing performance when adopting parallel architectures, and that we might expect even better results with the use of large clusters with many processing nodes.

1. Introduction

Professionals in any sector need to have access to accurate and complete knowledge to be able to take well-informed decisions. This is getting more and more difficult due to the sheer size of data they need to process. This also means that the knowledge and information of professionals is quickly getting out of date. However, their decisions have an even bigger impact in today's highly-interconnected world. Thus, professional decision-makers are involved in a constant race to stay informed and to respond adequately to any changes, developments and news. However, the volume of news and documents provided by major information brokers has reached a level where state-of-the-art tools are no longer adequate to provide a solution.

Processing huge amounts of textual data has become a major challenge in the Natural Language Processing (NLP) research area. As the majority of digital information is present in the form of unstructured data such as web pages or news articles, NLP tasks such as cross-document coreference resolution, event detection or calculating textual similarities often require processing millions of documents in a timely manner. For example, the main goal of the Newsreader project[1] is to perform multilingual real-time event detection and extract from text what happened to whom, when and where, removing duplication, complementing information, registering inconsistencies and keeping track of the original sources. The project foresees an estimated flow of 2 million news items per day and the complex linguistic analysis of those documents needs to be done in a reasonable time frame (one or few hours). Therefore, the project faces an important challenge with respect to the scalability of the text processing.

This overwhelming flow of textual data calls for a paradigm shift in the computing architecture and large scale data processing. For instance, Singh et al. [27] process a corpus comprising news articles published during the last 20 years. McCreadie et al. [20] present a distributed framework for event detection that is capable of effectively processing thousands of twitter posts every second. These challenges fall into a new class of the so called "Big Data" tasks, requiring large scale and intensive processing which have to be able to efficiently scale up to huge amounts of data [23,27,20].

This paper presents a new distributed architecture and technology for scaling up text analysis to keep pace with the rate of current growth of news streams and collections. We designed and deployed a complete chain of NLP modules within virtual machines (VMs). We also present the twelve NLP modules included

* Corresponding author.
E-mail addresses: rodrigo.agerri@ehu.es (R. Agerri), xabier.artola@ehu.es (X. Artola), zuhaitz.beloki@ehu.es (Z. Beloki), german.rigau@ehu.es (G. Rigau), a.soroa@ehu.es (A. Soroa).

[1] http://www.newsreader-project.eu/.

http://dx.doi.org/10.1016/j.knosys.2014.11.007
0950-7051/

into the virtual machines, of which a subset, the IXA pipes tools, are the ones used to do most of the experimentation [1].[2] We also provide empirical performance results when applied to realistic volumes of news within hard time constraints.

The rest of the paper is organized as follows. Section 2 presents and discusses alternative big data frameworks. Section 3 presents the text analysis modules of the English and Spanish pipelines. Section 4 describes the distributed architecture and deployed solution for massive processing of streams of texts. Section 5 describes the experiments carried out to evaluate the performance of the proposed solution. Finally, in Section 6 we discuss some concluding remarks and planned future research work.

2. Big data frameworks

Processing massive quantities of data requires designing solutions that are able to run distributed programs across a large cluster of machines [30]. Besides, issues such as parallelization, distribution of data, synchronization between nodes, load balancing and fault tolerance are of paramount importance. *Apache Hadoop*[3] is a framework designed to perform large scale computations that is able to scale to thousands of nodes in a fault-tolerant manner. It is probably the most widely used framework for large scale processing on clusters of commodity hardware. Hadoop implements *MapReduce*, a programming model for developing parallel and distributed algorithms that process and generates large data sets. Hadoop is the basis for a large number of other specific processing solutions such as Mahout[4] for machine learning or Giraph[5] for graph processing, to name but a few.

One of the main problems of using the Hadoop framework is that it requires casting any computation as a MapReduce job. In the case of the NLP pipeline presented in this work, these solutions would require a complete reimplementation of each NLP module, which is clearly impractical. *Apache SPARK* [31] overcomes this problem by extending Hadoop with new workloads like streaming, interactive queries and learning algorithms.

Hadoop follows a *batch* processing model, where computations start and end within a given time frame. In a *streaming computing* scenario [8], however, the processing is open-ended. Thus, the program is designed to process documents forever while maintaining high levels of data throughput and a low level of response latency.

Storm[6] is an open source, general-purpose, distributed, scalable and partially fault-tolerant platform for developing and running distributed programs that process continuous streams of data. Storm is agnostic with respect to the programming model or language of the underlying modules, and, thus, it is able to integrate third party tools into the framework.

The main abstraction structure of Storm is the *topology*, a top level abstraction which describes the processing node that each message passes through. The topology is represented as a graph where nodes are processing components, while edges represent the messages sent between them. Topology nodes fall into two categories: the so called *spout* and *bolt* nodes. *Spout* nodes are the entry points of a topology and the source of the initial messages to be processed. *Bolt* nodes are the actual processing units, which receive incoming text, process it, and pass it to the next stage in the topology. There can be several instances of a node in the topology, thus allowing actual parallel processing.

The data model of Storm is a *tuple*, namely, each *bolt* node in the topology consumes and produces tuples. The tuple abstraction is general enough to allow any data to be passed around the topology.

In Storm, each node of the topology may reside on a different physical machine; the Storm controller (called *Nimbus*) is the responsible of distributing the tuples among the different machines, and of guaranteeing that each message traverses all the nodes in the topology. Furthermore, *Nimbus* performs automatic re-balancing to compensate the processing load between the nodes.

Section 4 describes our solution to big data processing using *virtualization*, Apache Storm and a set of NLP tools organized in a data-centric architecture. The description of such NLP tools is the subject of the next section.

3. NLP pipeline

Many Natural Language Processing (NLP) applications demand some basic linguistic processing (Tokenization, Part of Speech (POS) tagging, Named Entity Recognition and Classification (NERC), Syntactic Parsing, Coreference Resolution, etc.) to be able to further undertake more complex tasks. Generally, NLP annotation is required to be as accurate and efficient as possible and existing tools, quite rightly, have mostly focused on performance. However, this generally means that NLP suites and tools usually require researchers to perform complex compilation/installation/configuration procedures in order to use such tools. At the same time, in the industry, there are currently many Small and Medium Enterprises (SMEs) offering services that one way or another depend on NLP annotations.

In both cases, in research and industry, acquiring, deploying or developing such base qualifying technologies is an expensive undertaking that redirects their original central focus. In research, much time is spent in the preliminaries of a particular research experiment trying to obtain the required basic linguistic annotation, whereas in an industrial environment SMEs see their already limited resources taken away from offering products and services that the market demands. In order to address this issue, we have developed a set of NLP tools which we refer to as the IXA pipes tools [1].[7] The IXA pipes tools consist of *ready to use modules* to perform efficient and accurate linguistic annotation while allowing users to focus on their original, central task.

3.1. IXA pipes

The aim of the IXA pipes tools is to provide multilingual NLP tools that are simple and ready to use, portable, modular, efficient, accurate and distributed under a free license. As in Unix-like operating systems, the IXA pipes consists of a set of processes chained by their standard streams, in a way that the output of each process feeds directly as input to the next one. The Unix pipeline metaphor has been applied for NLP tools by adopting a very simple and well-known data-centric architecture, in which every module/pipe is interchangeable for another one as long as it reads and produces the required data format. The IXA pipes are designed to minimize or eliminate any installation/configuration/compilation effort and are distributed under the Apache 2.0 license, which is free and commercially friendly [1].

The data-centric architecture of the IXA pipes relies on a common interchange format in which both the input and output of the modules needs to be formatted to represent and filter linguistic annotations: the NLP Annotation Format (NAF[8] [16]). NAF has evolved from the KYOTO Annotation Framework (KAF [6]) and it is

[2] http://ixa2.si.ehu.es/ixa-pipes.
[3] http://hadoop.apache.org/.
[4] https://mahout.apache.org/.
[5] http://giraph.apache.org/.
[6] http://storm.incubator.apache.org/.
[7] http://ixa2.si.ehu.es/ixa-pipes.
[8] http://wordpress.let.vupr.nl/naf/.