Design and Modeling of Topic/Trend Detection System
By Applying Slow Intelligence System Principles

Ji Eun Kim, Yang Hu and Shi-Kuo Chang
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
{kim, huyang, chang}@cs.pitt.edu
and
Chia-Chun Shih and Ting-Chun Peng
Institute for Information Industry, Taiwan, Republic of China
{chiachun, markpeng}@iii.org.tw

Abstract

This paper first describes the characteristics of the Slow Intelligence System (SIS), and then provides an example of Petri Net modeling of the topic/trend detection system by applying the SIS principles. The Petri Net model is simulated using PIPE3 tool. Based upon the requirements for Slow Intelligence Systems, a survey of existing development framework leads to suggestions for a generic framework for SIS system development.

Keywords

Component-based software system, slow intelligence system, Petri net modeling, slow intelligent system, SIS framework

1 Introduction

With a trend of mass use of social media such as blogs, and social networking in Internet, the detection of trends and hot topics out of millions of heterogeneous data become important not only to the users of the systems but also commercial stakeholders around these kind of systems. For example, HP Labs have demonstrated that social media can be effective indicators for predicting movie revenues [2]. Most of social medial platform providers adopt the technologies from Information Retrieval research to provide customized and the most relevant services and advertisements to the users. However, the data generated by users through social media is growing exponentially, this requires new approaches to collect and process many heterogeneous data available on the web efficiently. The new approaches should reduce the search space by intelligently selecting the most relevant data sources on the web, and adapt data processing algorithms to handle varieties of the data automatically or semi-automatically [15].

A Slow Intelligence System (SIS) [16] is a general-purpose system characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration, which are the characteristics of the Slow Intelligence System. A SIS continuously learns, searches for new solutions and propagates and shares its experience with other peers. A SIS differs from an expert system in that the learning is not always obvious. A SIS seems to be a slow learner because it analyzes the environmental changes and carefully and gradually absorbs that into its knowledge base while maintaining synergy with the environment. A Slow System in general has two decision cycles – a quick decision cycle providing an immediate response to the environment and a slow decision cycle that tries to follow the gradual changes in the environment by analyzing the information acquired from experts and past experiences so that the performance of the system is improved over time.

The online topic/trend detection system in [15] adopted Slow Intelligence for their design so that computing resources are gradually concentrated on prospect solutions. We employ their design approaches and model an online topic/trend detection system in Petri Net model. In addition, we recommend the SIS generic framework, which integrates the subsets of selected existing software frameworks in order to support the entire lifecycle of the SIS.

The remainder of this paper is organized as follows. Section 2 illustrates Petri Net modelling of topic/trend detection system as a case study and shows the simulation results. Section 3 provides a survey of existing software frameworks with respects to requirements for SIS systems. Finally, recommendations for the generic SIS framework are given in Section 4.

2 Modelling of Topic/Trend Detection System in SIS

The online topic/trend detection system (TDT) proposed by [15] is to detect current hot topics and to predict future hot topics based on data collected from the Internet. Since it is unlikely to collect all data on the Internet, the system requires users to provide their information needs, including their concerned keywords and their concerned websites.
Furthermore, since hot topics change quickly, the system requires periodical updates in hourly or daily intervals. The system first collects latest data from Internet based on users’ information needs by Crawler & Extractor, then adopts TDT techniques to discover current hot topics by Topic Extractor, and finally applies trend estimation algorithms [3] to predict hot topics by Trend Detector.

The Crawler & Extractor building block is for the restriction of web data in the topic trend detection. It is modelled as SIS system principle. Currently Topic Extractor building block is a black box component, which can be later modelled and refined. Trend Detector building block is modelled as SIS system.

2.1 Crawler & Extractor

The responsibility of Crawler is to collect web pages from Internet. Crawler needs to be selective, that is, only collect web pages that satisfy predefined requirements. Extractor is responsible to extract information from web pages.

Crawler

Several approaches have been proposed in the literature to restrict the media resources to crawl. The approaches include focused crawling [5], contextual crawling, semantic web and genetic-based crawling [7]. We observed that most of the existing approaches are close to the core concept of the SIS, which can be modelled as enumerator, adaptor, eliminator and concentrator.

We select the focused crawler as a basic model for the crawlers in the SIS in this project. However, the focused crawler can be replaced with another crawler depending on the different requirements and constraints for the target topic trend detection system.

The focused crawler consists of two phases: classifier and distiller. The first state of the classifier is to come up with the most relevant URLs for the topic trend detection. The list of URLs from the focused crawler can be different for each user because the SIS interacts with each user to enumerate example URLs and then recommend relevant URLs to the user. Then the focused crawler in SIS trains the knowledge base to get the most relevant URLs. In other words, we can map the classification phase of the focused crawler approach to enumerator and adaptor. Once training is done, SIS will identify the most relevant hubs by running a topic distillation algorithm. The results of this run raise the priorities of hubs and immediate neighbours. This distillation phase is mapped to eliminator in the SIS concept. The last phase is to report the most popular sites and resources to the users. The users finally mark the results as useful or useless, and send feedback to classifier and distiller. Thereby classifier and distiller can concentrate their crawling data. This whole cycle of enumerator, adaptor, eliminator and concentrator can be repeated.
Figure 3. Petri Net of SIS

The extractor is also modelled in SIS. Enumerator can include building the HTML tag tree. Adaptor mines data regions. A data region is a collection of two or more generalized nodes, which have the same parents nodes and are adjacent. The regions are found using tree edit distance algorithm. Identifying the data records from data regions can be a role of eliminator. Learning the structure and extracting of the data can be done in the concentrator in SIS. For each data region we need to understand the structure of the data records in the region. Therefore, partial tree algorithm [19] is used to gather the structure. Extraction of data from single and multiple pages often introduces a lot of noise (e.g., advertisement and non-relevant data). During the concentration phase, a sample page is taken as the wrapper. Then this wrapper is refined to solve mismatches. The literature [19] found different types of mismatches, which include text, string mismatches and tag mismatches. By using partial alignment and wrapper solution, the different structures are merged and build a generic template for a data record. Finally the extracted data is stored in the web data as a knowledge base.

2.2 Topic Extractor

The responsibility of Topic Extractor is to detect hot topics from a set of text documents. The process of topic detection can be divided into the following steps, which adopt some state-of-the-art techniques: 1) topic word extraction: TF-IDF [14] scheme is applied to measure the importance of terms in a given text document and generates top-N topic word candidates for each text document. 2) topic word clustering: single-pass clustering [1, 17], a popular topic detection approach, is adopted to cluster related documents into associated topic groups. The centroid topic word of cluster with highest weighting score is treated as the representative name of each generated cluster, which represents an extracted “topic”. 3) extract hot topics: hot topics derive from hot events in a particular timeline [1, 6].

The Topic Extractor is a black box component in our model. It connects Crawler & Extractor to Trend Detector.

2.3 Topic Detector

The responsibility of trend detector is to detect trends (future hot topics) based on currently available data. Figure 4 illustrates the subcomponents of the topic detector component, which consists of user request, SIS based TDT, knowledge base and dispatcher. Figure 5 illustrates how SIS principle applied to the topic detector. We model each subcomponent in Petri net.

User request handler

This subcomponent (Figure 6) is responsible for collecting the entire user request from the web server then sends the collected data to the TDT component.
Topic Detection and Tracking

TDT component (Figure 7) is responsible for finding the trend in the stream of user requests, and makes decisions in order to tune up the server cluster. Examples include sending the reducing power supply instruction to the group of servers that working on one topic when TDT component detect that this topic may cold in the future [8].

a. Slow Intelligence System

SIS is the fundamental framework for TDT who is going to use different methods to predict the trend. With its help, TDT can gradually improve performance and accuracy over time.

The input to this framework is the user request, which will be treated as the criteria of the current hot topic. Then the eliminator will calculate the current hot topic using different method’s combination generated by the enumerator and compare the result with the user request. Finally, the concentrator will select the best combination according to the result generated by the eliminator to help SIS in predicting the trend.

In the meantime, the intermediate result will be kept in two knowledge bases.

b. 1st Knowledge Base

1st KB (Figure 8) is the core database, which contains two knowledge bases that can help SIS to select the best method over time.

1) Evaluation result knowledge base
   The result about what method we have choice last time.

2) Sampled historical record knowledge base

2nd KB (Figure 9) saves the record for previous predicted trend and the actual hot trend at that time. So that the Eliminator can pick up the method and compare with these records to see which algorithm can generate the best result.

Dispatcher

Dispatcher gets the instruction from the TDT server and the topic (marked with hot or cold), then sends the associated instruction to the group that dealing with this topic.

2.4 Composition of other components

Since the topic/trend detection systems are sequentially ordered with Crawler &Extractor, Topic Extractor and Trend Detector, the output from the Petri-Net model of the crawler and extractor is the input to the Topic Extractor component.

2.5 PIPE 3 Simulation Results

PIPE 3 [13] is an open source, platform independent tool for creating and analyzing Petri nets. We employ PIPE 3 and extend it to support timing model. Our extended PIPE 3 enables us to specify immediate execution, which is shown as -1 in Execution Time in simulation results or the static time for the component execution.

<table>
<thead>
<tr>
<th>Name</th>
<th>Execute Time</th>
<th>Script Path</th>
<th>Source Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>-1</td>
<td>D:\DropBox\T0.jar</td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>-1</td>
<td>D:\DropBox\T10.jar</td>
<td></td>
</tr>
<tr>
<td>T11</td>
<td>-1</td>
<td>D:\DropBox\T11.jar</td>
<td></td>
</tr>
</tbody>
</table>
Figure 10. PIPE3 Simulation Results

3 Towards a Framework for Slow Intelligent System Development

As a part of the development of this novel software design methodology, we define a general framework for developing Slow Intelligence Systems. This paper discusses the development time and runtime aspect of SIS framework.

3.1 Requirements for SIS generic framework

We identify the six important requirements for the software frameworks for Slow Intelligent Systems (SIS) as follows:

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Java support is required to provide interoperability amongst different hardware and operating systems. Note that SIS shall provide different levels of interoperability by employing web services for distributed systems later. However, the current SIS framework requires platform independent environment to run components without supporting web services.</td>
</tr>
<tr>
<td>R2</td>
<td>Dynamic lifecycle management of building blocks during runtime is required because SIS principle allows a software component to be created, modified and removed dynamically.</td>
</tr>
<tr>
<td>R3</td>
<td>XML message based communications. The SIS components use XML messages which specify input and output components for each message type.</td>
</tr>
<tr>
<td>R4</td>
<td>IDE provides Code frame generation out of meta-models in a form of class diagram.</td>
</tr>
<tr>
<td>R5</td>
<td>Scalability of components up to 1,000 -10,000 components</td>
</tr>
<tr>
<td>R6</td>
<td>Coherent Integrated Development Environment (IDE) support to facilitate seamless development, simulation and deployment of the software components</td>
</tr>
</tbody>
</table>

3.2 Survey of existing frameworks

This section discusses the existing frameworks from development environment to runtime framework, as well as the relevant concepts for solving similar problems.

3.2.1 OSGi framework

The OSGi (Open Service Gateway Initiative) [12] service platform is a service platform for the Java programming language that implements dynamic component model. The bundle (software package with meta information) life cycle management such as start, stop, install and update is done via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly. The OSGi specification that defines a dynamic component system for Java enables a development model where applications are dynamically composed of many different reusable components. The OSGi specifications enable components to hide their implementations from other components while communicating through services, which are objects specifically shared between components.

SIS can employ many interesting concepts from OSGi for the SIS system. The software component life-cycle management (e.g., install, start, stop, update software bundles) feature can be a solution for our requirement of
R2 (Dynamic lifecycle management of software building blocks during runtime) and OSGi framework is directly usable to the SIS framework. Distributed EventAdmin service is applicable to realize the requirement of R3 (XML message based communication). In addition, OSGi’s dynamic service handling concept and declarative service concept can be employed to SIS framework in order to support R2.

3.2.2 Eclipse EMF

EMF [10] is an Eclipse plug-in, which supports the java code frame generation based on Ecore model, which is a UML representation of classes. Once users specify an EMF model, the EMF generator can create a corresponding set of Java implementation classes. Users can edit these generated classes to add methods and instance variables and still regenerate from the model as needed: users’ additions will be preserved during the regeneration. If the code the user added depends on something that users changed in the model, users will still need to update the code to reflect those changes; otherwise, users’ code is completely unaffected by model changes and regeneration.

Therefore, it can support R4 (Code generation) and can be used together with Eclipse IDE. In order to make EMF useful, the SIS model should be available in a form of Ecore model or other supported UML formats such as Rational Rose.

3.2.3 Eclipse ECF

Eclipse ECF (Eclipse Communication Framework) [9] is an Eclipse plug-in, which provides a communications container. ECF containers represent access to a protocol-specific communications context. ECF can be integrated into SIS framework’s communication concept realization. ECF supports both point-to-point and publish-and-subscribe communication methods, which can be important realization of SIS’s message based communication for the distributed building blocks. ECF generic container can be usable for the SIS domain specific message contents.

3.2.4 ACME

ACME [8] is a software architecture description language to support software architecture, design and analysis. It provides an architectural ontology consisting of seven basic architectural design elements, a type mechanism for abstracting common, reusable architectural idioms and styles and an open semantic framework for reasoning about architectural description. ACME language and the Acme Studio tool are possibly integrated into the SIS framework. ACME supports the extensibility of the domain specific architectural styles. SIS may provide a catalog of architectural styles, which are commonly used for the SIS components. Depending on what properties we want to assure during the design time, we can extend ACME to enable SIS specific analysis.

3.2.5 CASCADA

CASCADA [11] aims to provide an autonomic component-based framework to support the deployment of a novel set of services through development of distributed applications capable of coping well with uncertain environments by dynamically adapting their plans as the environment changes in uncertain ways. To enable dynamic software component/services management over time, their architectural concept to support plan, execution, and self-* model can be adaptable to SIS purpose.

Figure 11 illustrates the relationships between different technologies and how each technology can be applied to the SIS framework. We envision developing SIS systems using Eclipse IDE. The core features of the SIS
framework shall be realized by extending existing technologies, as Eclipse plug-ins. SIS systems are compliant to SIS architectural styles, which is extendable in ACME. The meta model for SIS components shall be represented using Eclipse EMF. Dynamic lifecycle management of SIS components can be realized by using OSGi framework. SIS system supports message based communication through Eclipse ECF. The existing PIPE 3 tool shall be also ported to an Eclipse plug-in. The SIS approaches may adapt CASCADAS’s self-* model to add context-aware decision making mechanism in addition to the current SIS principle.

4 Conclusion

We recommend the SIS generic framework use Eclipse as an IDE and build the SIS software framework by employing existing framework such as OSGi. For the robust development environment support, we need more research on the meta-model description for the SIS, and integrate the meta-model to the target SIS architecture. To support architectural design, we recommend adding SIS architectural styles to the existing ACME. The concept of the CASCADAS project is also recommended to take consideration to support decision-making process of SIS. In addition, implementing the PIPE 3 tool as an Eclipse plug-in can provide seamless and coherent development, simulation and deployment environment by leveraging Eclipse ecosystem.

Acknowledgements

This study is conducted under the "Innovative and Prospective Technologies Project" of the Institute for Information Industry, which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

[12] OSGi Alliance, Available at: http://www.osgi.org