2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)

Natural Language Processing and Text Mining to Identify Knowledge Profiles for Software Engineering Positions

Generating Knowledge Profiles from Resumes

Rogelio Valdez-Almada, Oscar M. Rodriguez-Elias, César Enrique Rose-Gómez, María De Jesús Velázquez-Mendoza
División de Estudios de Posgrado e Investigación
Tecnológico Nacional de México - Instituto Tecnológico de Hermosillo
Hermosillo, Mexico
ing.rvaldez@gmail.com, omrodriguez@ith.mx, crose@ith.mx, rvelazqu@ith.mx

Samuel González-López
División de Estudios de Posgrado e Investigación
Tecnológico Nacional de México - Instituto Tecnológico de Nogales
Nogales, Mexico
samuelgonzalezlopez@gmail.com

***Abstract*— Organizations frequently report problems finding skillful people to cover their most knowledge intensive vacancies. Being software engineering positions some of the such kind of jobs, there is a considerable gap between job postings and hiring skillful engineers in many software engineering organizations. In this paper, we will introduce the prototype of a web application that helps identifying Technical Knowledge (TK) in software development, to serve as a tool in the hiring process of software engineering positions, and in talent management. The purpose of this tool is to do an initial screening when opening a job position. All this is accomplished using Natural Language Processing (NLP) and Text Mining (TM) to analyze unstructured text in resumes and curriculum. We propose a way to use NLP and TM to identify knowledge profiles for Software Engineering Positions.**

Keywords—Natural Language Processing; Text Mining; Software Engineering; Knowledge Profile;

I. INTRODUCTION

Technology is becoming more important in our daily routine. Probably the most obvious example are computers (e.g. laptops, smartphones, tablets, smart watches and many other gadgets) and the internet, which are essential in many (if not every) activities we do. All of the former need software (programs) that tell these devices what to do, when to do it, how to work, etc., therefore, there is an increase in the demand for software engineers in making (also maintaining and expanding) the programs for these devices.

According to [1], it is estimated that software engineering will have a 17% to 19% growth by 2024. Also, [2] stares that there is a gap that is just over 72% between job postings and hires in their job hiring platform. This shows us that for some reason, there are a lot of job postings that are not covered. Although some of this postings may be covered elsewhere and not through them, even if half of the postings are covered elsewhere e.g. covered directly by companies, other job platforms or the position is cancelled, etc., we are talking about a gap of 36% of software engineer positions without hires.

This paper introduces the idea of using Natural Language Processing (NLP) and Text Mining (TM) in order to help reducing this problem, by analyzing and processing resumes to detect Technical Knowledge (TK) while saving time and effort. This leads us to a software prototype called the Knowledge Profile Generator (KP GENERATOR). KP GENERATOR was conceived with the purpose of helping in the identification of TKs of candidates for software engineering positions. The KP GENERATOR creates a knowledge profile that shows each of candidate's TKs. Using an application that combines NLP and TM as the KP GENERATOR will enable companies to not only know the TK of the possible candidates, but also to know the TK that may now be part of itself.

Another advantage of KP GENERATOR is that the Human Resources (HR) team can focus on other parts of the selection process such as interviews or tests. Using the KP GENERATOR could help improve the selection process in companies making it more efficient and less time consuming, especially when the requirements are specific technical knowledge skills needed for a position, i.e. knowledge intensive positions.

KP GENERATOR can process unstructured text that contains TK in software engineering. This unstructured text can be a Curriculum Vitae (CV), a resume or any document of this nature (having TK in software engineering positions). For the rest of this paper we will refer to this documents as "resumes" to refer to any document of this nature that can be used as input for the system. KP GENERATOR may be referenced in different ways e.g. when a reference is made to "the prototype", "the system", "the application", etc. we are referring to the KP GENERATOR prototype without trying to make any distinction between one or another of the terms mentioned. The reason for

0-7695-6342-2/17/31.00 ©2017 IEEE
DOI 10.1109/CONISOFT.2017.00019

using other terms depends solely on the context of what is being described.

In this next section we will describe the problem and issues involving HR, software engineering and knowledge intensive workers. In the third section we will discuss the related work. In the fourth section we will describe the tools used in the creation of the KP GENERATOR prototype. The fifth section is about the implementation of the KP GENERATOR. The sixth section describes the results of the prototype, to finally conclude in section seven.

II. Difficulties in hiring knowledge intensive workers

Human Resources (HR) is the department in a company in charge of the hiring process, which is a long process concerning several stages including the recruitment and selection processes. The personnel in HR may have the expertise in software engineering, but they still need to spend a lot of time looking at the resumes during the recruitment process. Investing this time in other parts of the recruitment process can make the whole process more efficient. Over the years, companies and HR have changed its main focus [3]; in the industrial era, companies focused on obtaining land and a large workforce. On the other hand, during the knowledge era the focus changed to business process improvement, as well as knowledge and talent management.

Talent management involves knowing the skills you have inside your company and taking care of them (e.g. performance management, promotions, compensations, etc.). With the former, we are referring to the people that have the skills and not the skills per se. The five elements of Talent Management according to [4] are the following:

- Recruitment and talent acquisition: Involves activities to discover, attract, evaluate and hire the talent needed for a company.
- Learning management: How a company manages each employee's development, education, certifications, etc. either to develop skills or to ensure compliance with any government or professional requirements.
- Employee performance management: A continuous process to measure and improve employee performance. These activities include: performance appraisals, goal management, multi-rater and 360-degree feedback.
- Workforce and succession planning: The process of planning and placing human capital resources to fill positions with motivated employees with the skills and experience needed for the position, as well as identifying and attracting candidates for present and future opportunities.
- Compensation management: Balancing employee performance and company goals with compensations given to the employee.

Knowledge workers according to [5] "have a high degree of expertise, education or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge". Wiig [6] mentioned knowledge intensive work and defined increased knowledge intensity as "a function of how much knowledge and understanding a person must process and apply when required to perform competent work and to be prepared to deal with uncertainties and surprises". Many software engineering positions, could be considered as knowledge intensive work since it involves having a large amount of background knowledge and a good understanding of the problem to be solved. KP GENERATOR focuses on technical knowledge which is part of knowledge intensive work.

KP GENERATOR focuses on finding this knowledge intensive workers, also on improving the recruitment and talent acquisition in talent management by reducing the time on the activities related to the discovery of talent needed of the company.

III. Related work

The combination of HR with NLP and TM have created an area of opportunity in the development of tools to enhance its performance and methods. Some works focus on students, candidates, for example, CaPaR [7] generates personalized job and skill recommendations for students by analyzing their profiles and resumes using TM and collaborative filtering techniques, this work focuses on the student and on the idea of recommending possible skills to develop. Another approach is to normalize CV (or resumes) structures like [8] A Framework for CV structuring which is an extension for General Architecture of Text Engineering (GATE) that analyses and structures the CVs normalizing CV content according to the structure adopted by Europass CV. Other works focus on the concepts and knowledge base. The work of [9] called SmartSearch focuses on generating a knowledge base and terminologies that is an update that reduces the gap between recruiters and candidates. The work of [10] uses text mining approach to classify employees using resumes in order to improve the resource utilization in their Resource Planning tool. KP GENERATOR focuses on the HR side for analyzing the resumes, but as an independent tool that provides information to the recruiters or HR department. KP GENERATOR does not look for a standardized way of writing resumes (or CVs). KP GENERATOR focuses on the creation of knowledge profiles by analyzing raw text in English in the area of Software Engineering. All this, using NLP and TM techniques.

IV. Theoretical background

A. Knowledge Profile

There is no general consensus on the definition of knowledge profile. In this paper, we will define knowledge profile according to [11] which is defined as "a set of structured features that describe the required knowledge, associated with resources and capabilities, which enable the dynamic generation of work competencies, key processes competencies and distinctive competencies that add value to the organization".

As mentioned before, KP GENERATOR creates a knowledge profile from the resumes. These knowledge profiles are not meant to be static and read-only, but a dynamic and modifiable base of a knowledge profile containing the TK of the candidate where competencies (work, processes or

distinctive) can be added as the company and HR team discover them.

KP GENERATOR creates the knowledge profile based on the ontology model created by [11]. This ontology model was used in the creation of the Knowledge Profiles Fuzzy Logic Valuation System (SVDPC in Spanish) [12] which consists of three modules which are the following:

- Profiling: This module contains everything related to the definition of the knowledge profile for a position.
- Evaluation: This module is related to the measurement of the knowledge level of a candidate for this position.
- Valuation: The main module which makes the comparison of the knowledge profile for the position and the knowledge profile obtained from the evaluation.

KP GENERATOR focuses on complementing the evaluation module of the SVDPC by adding a profile with the technical knowledge that the candidate has according to his/her resume. KP GENERATOR is based on the SVDPC model for compatibility and support of the latter to help with the capturing of the knowledge profile of the candidate instead of inserting all the data from zero and analyzing the resume manually.

B. Stanford CoreNLP: Natural Language Processing Library

In the creation of the KP GENERATOR prototype we used Stanford CoreNLP library version 3.7.0 [13]. CoreNLP is "an integrated suite of natural language processing tools […] including tokenization, part-of-speech tagging, named entity recognition, parsing, and coreference". One of the reasons for using CoreNLP is that it is, to our knowledge, the most complete, open-source NLP library available. This suite consists of different annotators that transform the raw text to annotated text. The CoreNLP documentation specifies that some annotators' models were created using supervised machine learning while others are rule-based. The main CoreNLP annotators used in the KP GENERATOR prototype are the following [13]:

tokenize: Tokenizes the text, stores the character offset and helps handling "reasonable noisy and web text"; tokenization can be considered as the first step in the whole NLP process.

ssplit: Splits tokenized text into sentences.

pos: Labels tokens with their Part of Speech (POS).

lemma: Generates de base forms (lemmas) of all tokens.

ner: Stands for Name Entity Recognizer (NER) which helps identifying entities such as persons, places, others (misc), as well as numeric entities e.g. money, date, time, etc.

regexNer: Regular Expression NER allows user to incorporate custom annotators using simple rules that may not be present in the CoreNLP corpora but are easy to recognize.

Stanford TokensRegex [14] is a framework included in Stanford CoreNLP for defining patterns over tokens. The advantage of using regular expressions in NLP is that you search for patterns over the tokens (including their attributes and tags) instead of working "manually" over the text.

C. Text Mining in the KP GENERATOR prototype.

Text mining, according to [15] is related to data mining because "data mining is about looking for patterns in data and text mining is about looking for patterns in text" this means that they share some techniques and methods for finding patterns. In [15], text mining is defined as "the process of analyzing text to extract information that is useful for particular purposes". In our case, the purpose is to create knowledge profiles from unstructured text (resumes), and to help the HR departments in companies on the hiring process of job opening related to software engineering position.

It is known that data mining and text mining share common techniques and approaches. From a general perspective, text mining follows the general model of data mining systems dividing its parts into four main areas: preprocessing tasks, core mining operations, presentation layer components and browsing functionality, and refinement techniques [16].

The general process of the KP GENERATOR model is shown in Fig. 1, and described below.

- Preprocessing Tasks: all techniques, methods, processes or approaches related to the preparation for a text mining system's core knowledge discovery operations. This include converting each original data source into an accepted format before applying feature extraction methods to create a new collection of documents.
- Core Mining Operations: The main functions of a text mining system including, but not limited to, pattern discovery, trend analysis, and incremental knowledge discovery algorithm.
- Presentation Layer Components: The Graphic User Interface (GUI) and pattern browsing functionality as well as access to the query language. This layer includes tools for creating or modifying the cluster's criteria.
- Refinement Techniques: Methods that filter redundant information and cluster closely related data. May include more complex applications such as those that include suppression, ordering, pruning, generalization, and clustering approaches aimed at discovery optimization.

The KP GENERATOR system contains all these four text mining components. First, it prepares or preprocesses the text

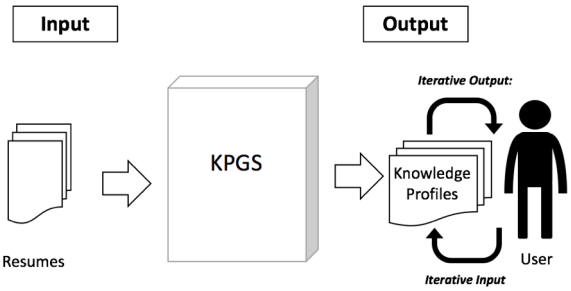

Fig. 1. KP GENERATOR text mining model. Based on [16] iterative loop for user input and output.

to a canonical structure for the NLP and Core Mining component. Then "mining" begins when the KP GENERATOR looks for the TK in the text and starts creating the knowledge profile of the candidate. The results of this process is shown in the GUI where the user interacts with the system. Finally, the post-processing where the user may filter or cluster data from the knowledge profiles to obtain the results according to his/her needs for a job position.

In this particular case for KP GENERATOR, the text mining technique used is text categorization (also named text classification) and a knowledge engineering approach is used. Text categorization is defined by [16] as "the task is to classify a given data instance into a pre-specified set of categories. Applied to the domain of document management, the task is known as text categorization – given a set of categories (subjects, topics) and a collection of text documents, the process of finding the correct topic (or topics) for each document" and mentions that knowledge engineering approach to text categorization "is focused around manual development of classification rules. A domain expert defines a set of sufficient conditions for a document to be labeled with a given category".

Where the given data are the resumes, the pre-specified categories are the TKs (and its categories), the domain is software engineering, and the process being to find the correct TKs for each resume. A document can belong to more than one category, i.e. a resume can have more than one TK (and therefore belong to different categories). The rules used in the knowledge engineering approach in the application will be discussed later in this paper.

D. Connecting CoreNLP server and the KP GENERATOR

The Stanford CoreNLP suite introduced CoreNLP server since version 3.6.0 at the end of the year 2015 [17]. Managing the natural language processing as a server provides advantages, specially by separating the NLP as an independent module from the rest of the application. Fig. 2 shows the Stanford CoreNLP server ready to receive requests, notice that this is an independent server from the rest of the application with its own listening port.

To connect to the CoreNLP server a modified version (which we will refer to as "new PHP adapter") of the PHP adapter created by Dennis de Swart [18] was used (which will be referred to as "original PHP adapter"). The original PHP adapter included the default annotators from CoreNLP and basic

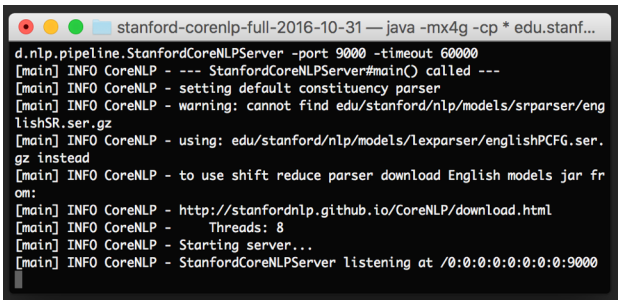

Fig. 2. Terminal showing Stanford CoreNLP server loading the models and ready for requests.

functionality. The new PHP adapter includes support for the TokensRegex annotator. This custom annotator was created and stored in the Stanford CoreNLP server, and the new PHP adapter had to suffer modifications in the request parameters to add the "technical" annotator which is the name of the custom annotator created.

The functionality for adding custom TokensRegex Annotator using a rules file was added in Version 1.3.2 in the year 2012 [19]. which was used to create the knowledge base and rules that find the TKs of the candidate. The requests to the CoreNLP server from the PHP adapter (both new and original) are made using cURL which used URL encoding. It is used to encode special characters in the URLs such as double quotes, commas, colons, etc. The elements that make up the knowledge base will be discussed later, in the following section.

V. MAKING THE APPLICATION

The creation of the KP GENERATOR prototype is based on the waterfall model. This means that there was a phase in which a minimum set of requirements were made to establish the needs to be solved by the application. Two main functional requirements were defined.

- The system must use text mining techniques to generate the knowledge profiles.
- The system must be able to generate the knowledge profile from unstructured data (text).

After having this requirements, the use case scenarios where developed. The use case diagram shows the interactions between actors (users and other systems) and the prototype (the system). Fig. 3 shows the use case diagram of the prototype, observe the interactions of the user with the system. Also notice that the SVDPC (mentioned in the previous section) can "visualize" results, i.e. this outside system can obtain the knowledge profile that was generated.

The four use cases are the following:

- UC01 – Input Text: This use case refers to everything related to the selection and the input of the resumes to the system, as well as the preprocessing and preparing of the resume before the processing.

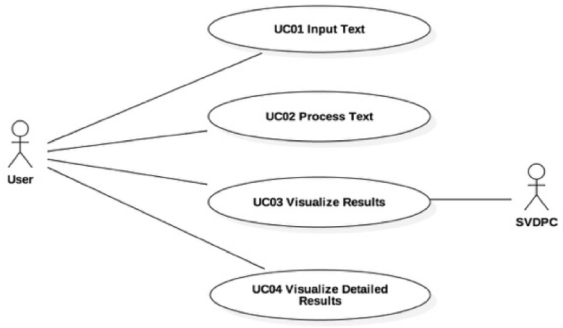

Fig. 3. Use Case Scenarios Diagram showing the functions for the KP GENERATOR prototype.

- UC02 – Process Text: This involves everything related to the creation of the knowledge profile like the Natural Language Processing of the resume and Core Mining Operations. The result of this processing is the knowledge profile.
- UC03 – Visualize Results: This use case represents all general visualization of the results from processing the resumes i.e. general visualization of the knowledge profiles.
- UC04 – Visualize Detailed Results: this refers to the specific visualization of any knowledge profile, the TKs that it contains as well as any other data it contains.

The system was modeled using Object Oriented Programming (OOP) paradigm. OOP was chosen for its advantages like being able to separate components (objects), its attributes, functions (methods), as well as having code that is easier to maintain. The architecture of the system we based on [20] which is based on the general architecture for TM systems proposed by [16], combined with Model-View-Controller (MVC) architectural pattern, to follow the Yii framework logic for implements web applications. MVC is used to divide an application (or part of it) in three components: the model, the view and the controller [21]. The controller contains the business logic and processes of an application The view shows information to the user and receives user input, and the model is responsible for the actual accessing of data. MVC patter enable each component to be loosely coupled, to minimize the impact of modifications and updates on the rest of the system's components.

The prototype was conceived as a web application for easy distribution and updating of the system. For that reason, KP GENERATOR is written in PHP. Other reasons to use this programming language include the experience of the developing team and for homogeneity, i.e. keeping the same programming language as the SVDPC. For the implementation of the MVC pattern the Yii2 Framework [22] was used. Yii2 is a PHP framework with the advantage of being fast, lightweight and including a code generator known as Gii.

A. Knowledge Base and Rules

As previously mentioned, KP GENERATOR was created using Stanford CoreNLP and Yii2 Framework. This two tools by themselves are not enough for creating the knowledge profile. There is a need for some previous knowledge that can define what are the TKs that the system is trying to find. If you recall, it was discussed that CoreNLP had some annotators that were trained (using machine learning) and others were rule-based. KP GENERATOR has a knowledge base that enables the system to find the TKs and to create the knowledge profile.

The prototype's knowledge base consists of a non-exhaustive list of TKs in software engineering, which can be observed in Fig. 4. This knowledge base was created for testing purposes for the prototype and should be expanded if more TKs in software engineering are needed. In the implementation of the prototype the rules were created for the Stanford TokensRegex (included in the CoreNLP suite) to annotate the TKs found in the input text (resumes). The TKs are classified in eight categories. Each category contains TKs that share similarities according to their characteristics. The categories are the following:

Programming and Markup Languages		Integrated Development Environments	Back End Frameworks	Front End Frameworks	Unit Testing Frameworks	Relational Database Management Systems	Non-Relational Databases	Data Management Frameworks
C	Objective C	Eclipse	.NET	AngularJS	Arquillian	Apache Hive	Apache Cassandra	Active Record
C#	Pascal	IntelliJ	Akka	Backbone	JUnit	BigQuery	BigTable	Entity Framework
C++	Perl	NetBeans	CodeIgniter	Boilerplate	KarmaJS	DB2	CouchDB	LINQ
COBOL	PHP	Visual Studio	Django	Bootstrap	Jasmine	MariaDB	MongoDB	Medoo
Groovy	Python	Xcode	Laravel	Ember	Mockito	Microsoft Access	Redis	SQLAlchemy
Haskell	Ruby		NodeJS	Foundation	NUnit	Microsoft SQL Server		
HTML	SmallTalk		Play	jQuery	PHPUnit	MySQL		
Java	Swift		Ruby on Rails	Kendo	PyUnit	Oracle		
JavaScript	Visual Basic		Yii	ReactJS	Selenium	PostgreSQL		
Mercury	XML		Zend	Semantic UI	VSUT Framework	SQLite		

Fig. 4. Non-exhaustive TK list for software engineering used in the KP GENERATOR prototype.

- Programming and Markup Languages: technical knowledge that are the languages used to write software applications.
- Integrated Development Environments (IDEs): technical knowledge of the use of this environment that are used to develop some specific kind(s) of applications e.g. Xcode for Apple apps (iOS, macOS, watchOS, etc.) and Visual Studio for developing applications based on Windows software.
- Back End Frameworks: technical knowledge related to frameworks for the business logic of the application and the processes that make its functionality.
- Front End Frameworks: technical knowledge related to frameworks involving the visual part of the applications, i.e. everything related to the views and user experience.
- Unit Testing Frameworks: technical knowledge related to the use of frameworks on the performing of testing applications.
- Relational Database Management Systems: technical knowledge related to using systems for managing databases.
- Non-Relational Databases: technical knowledge that are related to non-traditional databases (relational) are included in this category.
- Data Management Frameworks: technical knowledge that are related to the use of frameworks that help in the management of data and how it is accessed in the database.

The TK categories give a general picture of the kind of TKs that a candidate may have. For the moment, the KP GENERATOR prototype only looks for TKs that are explicitly written in the resume and does not make inferences, but the users may infer additional capabilities from the categories and TKs found. For example, the user can get the idea that a candidate is a Full-Stack Developer if he has strong TKs in Front End and Back End categories. The user (HR worker) may ask follow-up questions further on the selection process.

As you can see, Fig. 5 shows a fragment of the technical knowledge rules file used for the Programming and Markup Languages category. Notice how the programming language rule for "Objective C" needs to include alternative ways that the candidate may have written this TK in his/her resume. Another example of this kind of issues that may be present in the resumes is "Postgres" as an acceptable nickname for "PostgreSQL", where both refer to the same TK. As the former examples, there are many cases to consider when creating the rules for the knowledge base.

All these TK rules are in a rule's file, and this file has to be in the CoreNLP server (accessible for reading) when staring the server. Until this CoreNLP version (3.7.0), there is no endpoint for sending tokensregex rules file directly to the CoreNLP server, but only asking for the annotator that the server already has, which you may have created. Fig. 6 shows part of the new PHP adapter command for making requests to the CoreNLP

Fig. 5. Fragment of the Programming and Markup Languages category rules.

server notice that the custom annotator "technical.rules.txt" is called. That means that users cannot create their own rules for the CoreNLP server and send them to the server on the fly. The user would only be able to add new rules if you provide them access to the CoreNLP server directory and let them add the files, which obviously is not advised.

A knowledge profile should not only tell the TKs that a person has, but also how competent that persons is in each of the TKs. The KP GENERATOR plans to support levels of TKs by analyzing time references to the TKs in the future, but at the moment, all TKs found are assumed to be average level, i.e. all TKs found are regarded as 5 on the scale of 1 to 10.

B. How the Application Works

To implement and test the KP GENERATOR prototype a website was created for demonstration purposes to show the functionality of the system and its processes. Fig. 7 shows the main page of the system. Notice the tabs in the top which are used for navigation in the system. The Home tab shows the main page. The Demo tab takes us to the page where we can try the system. The CVs shows the resumes that are stored in the database. The Profiles tab shows all the knowledge profiles i.e. the connections relating the TKs with the resumes. The Categories tab shows the categories that are used to classify the TKs. The TK tab shows all the TKs that are stored in the

```
//  New customizable cURL Command with
//  Technical Knowledge annotator
define('CURLPROPERTIES',
'%22technical.rules%22%3A%22tokensregex/technical.rules
.txt%22%2C'.
'%22customAnnotatorClass.technical%22%3A%22edu.stanford
.nlp.pipeline.TokensRegexAnnotator%22%2C'.
'%22annotators%22%3A%22'.
'tokenize%2C'.
'ssplit%2C'.
'pos%2C'.
'parse%2C'.
'ner%2C'.
'regexner%2C'.
'technical%22%2C'.
'%22enforceRequirements%22%3A%22false%22%2C'.
'%22outputFormat%22%3A%22json%22%2C'.
'%22prettyPrint%22%3A%22true%22');
```

Fig. 6. The cURL property for the new PHP adapter wich is part of the query that is used to request data to the CoreNLP server.

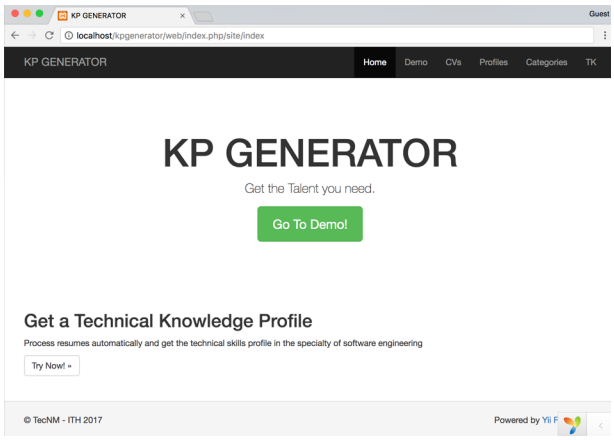

Fig. 7. The main view of the KP GENERATOR prototype.

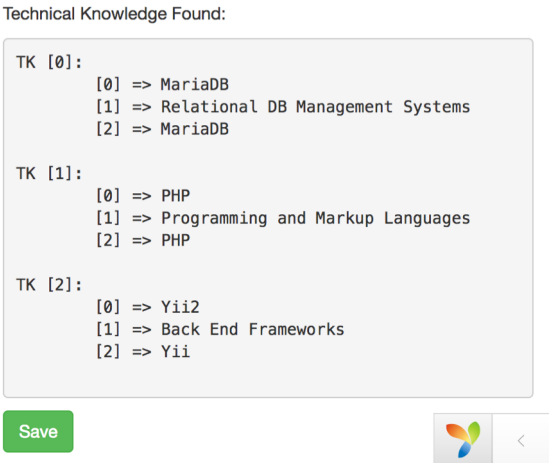

Fig. 9. KP GENERATOR showing the TKs found.

database. Notice that if the user wants to add a new TK, a new TK has to be added to the database as well as the rule for extracting this TK needs to be created to enable the system to find this new TK in the text it analyzes.

The KP GENERATOR prototype workflow starts with a view for the user to enter the text of the resume (Fig. 8) to analyze. Once entered, the text is preprocessed before sending the request to the CoreNLP server. Then, the NLP and TM begin, the resume is processed to assign it to a category (or categories) i.e. the resume is assigned TKs according to its content.

When the category assignment is completed, the finished product is the knowledge profile which contains the TKs in software engineering obtained from the resume. This finished product is shown in the view (Fig. 9) and the user may save the resume and profile in the database (Fig.10). Note that the reason

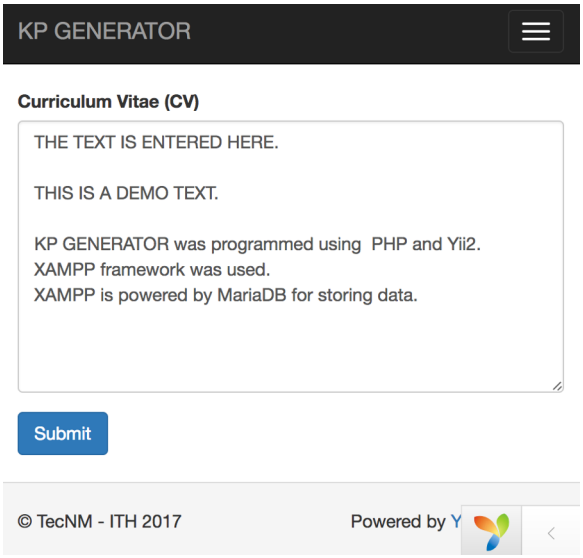

Fig. 8. Where the text is inserted by the user to start the processing.

for first showing the profile instead of saving it directly to the database is for demo and debugging purposes of this prototype. Finally, the user can explore the resumes in the database, view the knowledge profiles, select a profile to see its details, and filter the profiles according to the needs of the position in the company (Figs. 11-12).

VI. Application Results

For testing purposes of the KP GENERATOR prototype forty resumes were selected. This resumes were searched and taken from the web. These resume corpora are simulating the resumes of the candidates with experience in software engineering. Since they were taken from the web, some are resume templates containing an example of a resume with TKs to be mined, which is still useful for the purposes of testing the KP GENERATOR prototype and proving that knowledge profiles can be created using NLP and TM from unstructured text.

The first result to observe is the TKs that where obtained in the whole resume collection shown in Fig. 13. The most found

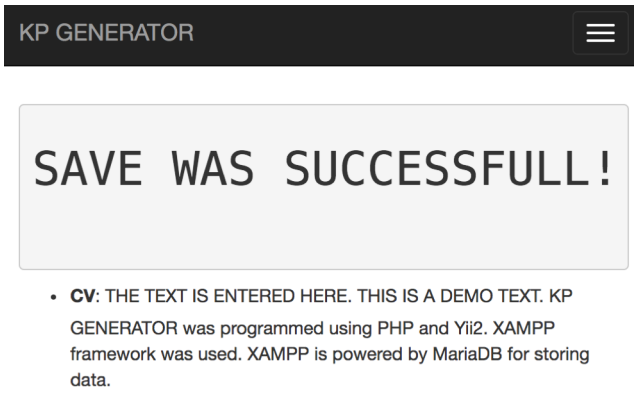

Fig. 10. The resume and knowledge profile are saved in the database and confirmation screen is shown.

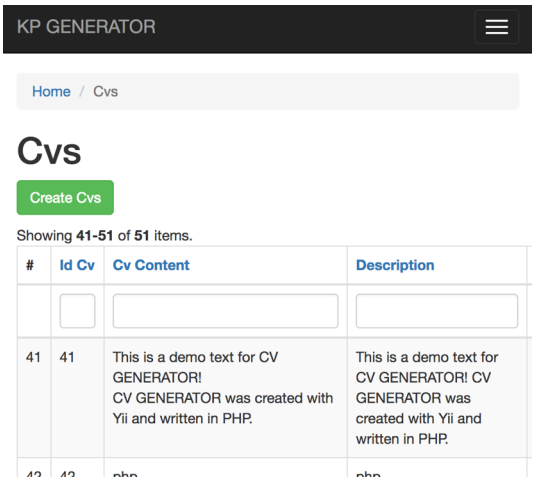

Fig. 11. View of the resumes in the database

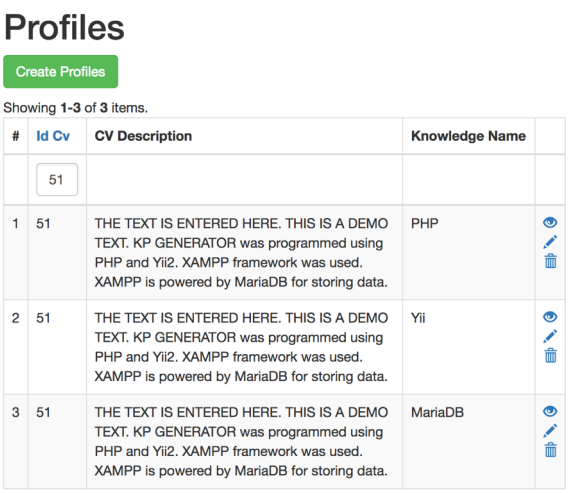

Fig. 12. View of the knowledge profile with id 51.

TK is Java which was found in 23 resumes, followed by JavaScript with a repetition rate of 20, HTML with 19 and PHP with 18. On the other hand, the least repeated (but found) TKs are IntelliJ, AngularJS, Backbone, Boilerplate and LINQ with a rate of 1 repetition; followed by Mercury, Ruby on Rails, Bootstrap, Foundation, Junit and PostgreSQL with a repetition of 2 and COBOL, Python, Ruby, and Zend with 3. Notice that not all the 75 TKs (Fig. 4) that are on the knowledge base appear on the graph of the technical knowledge found (Fig. 13). The TKs that are not shown on the graph are the TKs that have zero repetition on the resume collection used. There were 40 TKs with zero repetition according to the testing of KP GENERATOR. Further comparison will be made later on this section.

Notice that by analyzing the general results of the knowledge profiles for this simulation we already have an idea of i) the most common TKs, ii) the rarest TKs and iii) the inexistent TKs in the whole candidate population whose resumes where received. All this new information without even starting to analyze any knowledge profile individually. To find the candidates the recruiter needs, he has to enter which TK he is looking for and he will be prompted with the filtered list of the candidates that meet the criteria.

The size of the text analyzed (resume), the size of sentences (delimited by periods) in the tokenization and the size of the blocks of text sent to the server per request affect the memory and speed of the system. If a resume has a substantial amount of text and long sentences, the processing time will be higher (slower). On the other hand, a brief resume with short sentences will have a lower (faster) processing time.

The processing of the resume corpora was made using a conservative sixty-five words per request to the server, which is low number. The application was hosted using XAMPP v5.6.30 (an Apache distribution) on a MacBook Pro with a 2.5 GHz Intel Core i7 processor and 16 Gb of RAM and the CoreNLP server was hosted in this same computer assigning 4Gb of RAM (from the 16Gb) to the NLP server. Both servers where not dedicated i.e. they shared the resources between themselves and with the applications the computer was running.

The request size of sixty-five words was selected with the purpose of having the results in a slow/bad case scenario (not

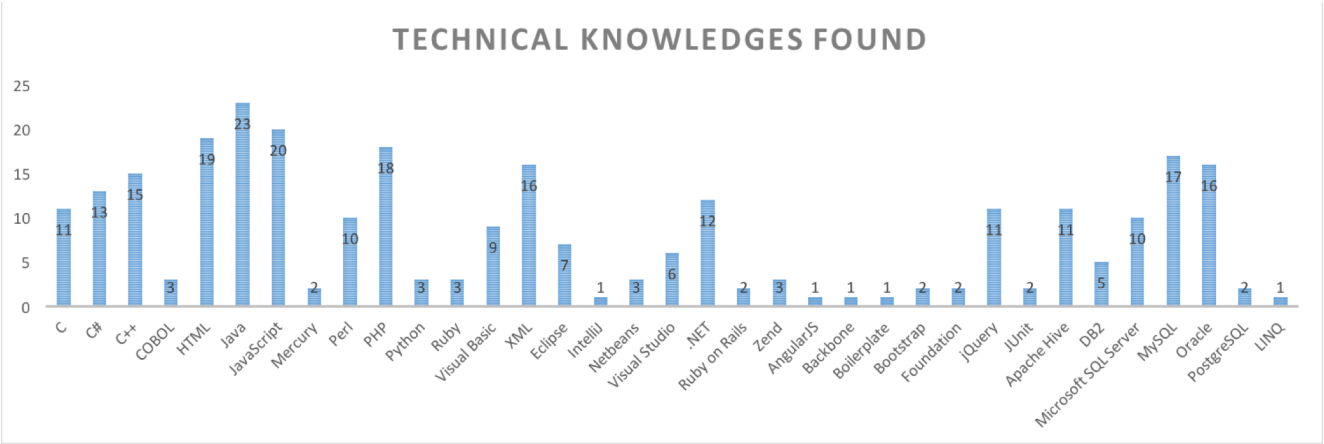

Fig. 13. The technical knowledge found on the resume corpora.

necessarily the worst case) and have the opportunity to adjust this set point in future testing. Note that choosing an extremely high request size can make the server run out of memory and crash or take too long and drop the request. A part of the results obtained while processing the resume corpora are shown in Fig. 14.

To compare the results obtained by KP GENERATOR a manual revision of the resumes was made. The manual revision was timed to compare processing time between KP GENERATOR and the time analyzing the resumes without this tool (Fig. 15). Observe that KP GENERATOR analyzed the resumes 3.89x faster than the manual revision. The processing time for each resume with both revisions (KP GENERATOR and the manual processing) can be observed on Fig. 16. The average speed of KP GENERATOR was 11.85 words per second and it took an average of 50.33 seconds per resume. On the other hand, the manual revision recorded a speed of 3.04 words per second with an average of 196.01 seconds per resume.

Resume Number	Processing Time (Seconds)	Requests To Server	Total Words	Words Per Second
1	45.27	9	636	14.05
2	45.41	9	523	11.52
3	25.15	5	369	14.67
4	25.16	5	330	13.12
5	55.32	11	730	13.20
6	95.59	11	806	8.43
7	40.24	8	622	15.46
8	65.37	13	912	13.95
9	75.58	15	1073	14.20

Fig 14. Fragment of the speed results obtained when analyzing the resume corpora.

	KP GENERATOR	Manual Revision
Total Processing Time (seconds)	2,013.16	7,840.41
Total Processed Words	23,846	23,846
Average Words Per Second	11.85	3.04

Fig 15. Total and average results of the testing of the resume corpora.

To measure performance, we used accuracy and F-measure, the latter is a typical value that combines precision (equivalent to positive predictive value) and recall (equivalent to sensitivity or true positive rate) basing on the work of [23]. β was given a value of 1 (for calculating F-measure). Giving β a value of 1 means precision and recall have the same weight. If β is greater than 1, recall is given more weight. The equations are the following:

- Precision, P = TP / (TP + FP)
- Recall, R = TP (TP + FN)
- F-measure = $((\beta^2 + 1)\ P\ R) / ((\beta^2\ P) + R)$
- Accuracy = (TP + TN) / (TP + FN + FP + TN)

Where: True Positive (TP) is where TK present in the CV and found by KP GENERATOR. False Positive (FP) is where TK found by KP GENERATOR but absent from the CV. False Negative (FN) is where TK present in the CV but not found by KP GENERATOR. True Negative (TN) is where TK absent from the CV and not found by NLP.

Comparing the results of the manual revision we were able to obtain the following results: there where 263 TP, 2680 TN, 19 FP and 39 FN found by the KP GENERATOR prototype. The accuracy measured in the testing of KP GENERATOR is 98.1% with a F-Measure of 90% (Fig. 17). The results show that the sensitivity or true positive rate (recall measurement) can be improved.

KP GENERATOR Performance	
Precision	0.932
Recall	0.870
F-Measure (β = 1)	0.900
Accuracy	0.981

Fig 17. KP GENERATOR performance values.

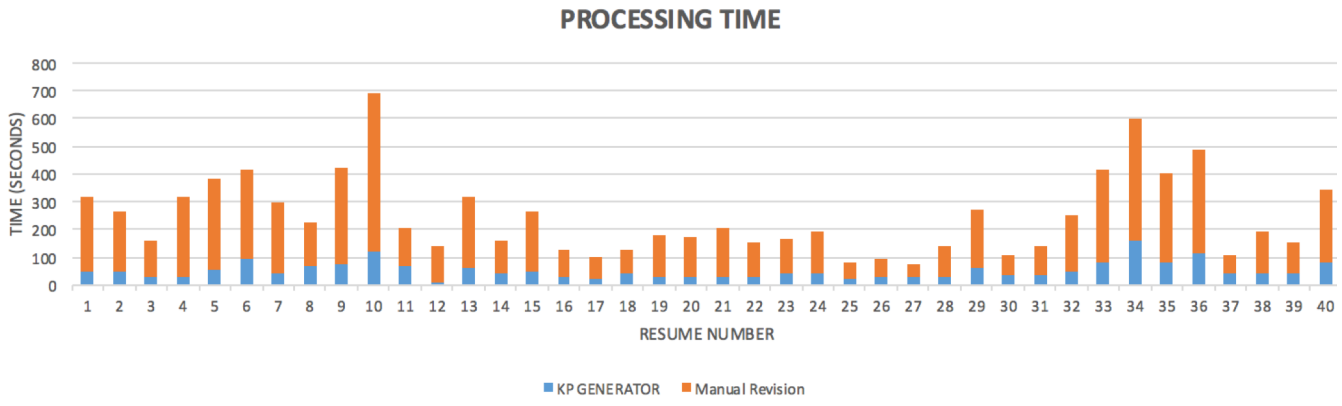

Fig. 16. Comparison of processing time of resumes.

VII. Conclusions

In this paper we proved that it is possible to identify the knowledge profiles using natural language processing and text mining for software engineering positions. This was done by implementing the KP GENERATOR, an application that was able to successfully extract the technical knowledge of the resumes analyzed and to create knowledge profiles from them. We were able to prove that this system can provide useful information that can help solve the problem of hiring people software engineering for knowledge intensive positions.

The results of the tests, showed us that the KP GENERATOR can significantly reduce the time HR recruiters expend reviewing resumes to identify the technical knowledge of the candidates. Moreover, we are certain that having the system on a dedicated machine with more computing power can significantly reduce the time even more, and make the system be able to analyze more resumes in less time.

Although the results we have obtained are promising, it is required to perform further testing, in order to analyze how much our proposal can reduce the effort made by HR departments of organizations while looking for experts to fulfill an important knowledge intensive job position. Additionally, we are also engaged on improving the KP GENERATOR by including other functionalities, such as defining a way to identify the expertise level by analyzing specific data on resumes, such as the time people has been working with specific technical knowledge.

Acknowledgment

We want to acknowledge the "Consejo Nacional de Ciencia y Tecnología" (CONACYT) for the contribution for the development of this research through the scholarship 708292/585476, given to the first author.

References

[1] M. Ward, "The 9 most in-demand jobs of 2017," 2017. [Online]. Available: http://www.cnbc.com/2017/03/27/the-9-most-in-demand-jobs-of-2017.html%0D. [Accessed: 15-May-2017].

[2] D. Hartley, "These are the most in-demand jobs for 2016," 2015. [Online]. Available: http://www.careerbuilder.com/advice/these-are-the-most-indemand-jobs-for-2016. [Accessed: 15-May-2017].

[3] I. Chiavenato, *Gestión del Talento Humano*. México: Mc Graw Hill, 2009.

[4] B. Docherty and M. Wasdin, *Talent Management For Dummies*. Wiley Publishing, Inc., 2007.

[5] T. H. Davenport, *Thinking for a Living: how to get better performance and results from knowledge workers*. Harvard Business School Publishing, 2005.

[6] K. Wiig, *People-Focused Knowledge Management : how effective decision making leads to corporate success*. Elsevier Inc., 2004.

[7] B. Patel, V. Kakuste, and M. Eirinaki, "CaPaR: A Career Path Recommendation Framework," *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, San Francisco, CA, USA, pp. 23–30, 2017.

[8] S. Amdouni and W. B. abdessalem Karaa, "Web-based recruiting," *ACS/IEEE International Conference on Computer Systems and Applications - AICCSA 2010*. IEEE, Hammamet, Tunisia, pp. 1–7, 2010.

[9] E. Malherbe and M.-A. Aufaure, "Bridge the terminology gap between recruiters and candidates: A multilingual skills base built from social media and linked data," *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, San Francisco, CA, USA, pp. 583–590, 2016.

[10] T. Gonzalez, P. Santos, F. Orozco, M. Alcaraz, V. Zaldivar, A. D. O. Obeso, and A. Garcia, "Adaptive Employee Profile Classification for Resource Planning Tool," *2012 Annual SRII Global Conference*. IEEE, San Jose, CA, USA, pp. 544–553, 2012.

[11] M. de J. Velazquez Mendoza, "Construcción de un modelo para el diseño de perfiles de conocimiento," Instituto Tecnológico de Hermosillo, 2013.

[12] J. A. Rosas Daniel, "Construcción de un modelo de lógica difusa para la validación de perfiles de conocimiento de personal," Instituto Tecnologico de Hermosillo, 2015.

[13] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," *Proc. 52nd Annu. Meet. Assoc. Comput. Linguist. Syst. Demonstr.*, pp. 55–60, 2014.

[14] A. X. Chang and C. D. Manning, "TokensRegex: Defining cascaded regular expressions over tokens," 2014.

[15] I. H. Witten, Z. Bray, M. Mahoui, and B. Teahan, "Text mining : A new frontier for lossless compression," 1999.

[16] R. Feldman and J. Sanger, *The Text Mining Handbook*. New York, United States: Cambridge University Press, 2007.

[17] Stanford CoreNLP, "Release History," 2017. [Online]. Available: https://stanfordnlp.github.io/CoreNLP/history.html%0D. [Accessed: 15-May-2017].

[18] D. Swart, "PHP adapter for Stanford CoreNLP," 2016. [Online]. Available: https://github.com/DennisDeSwart/php-stanford-corenlp-adapter.

[19] Stanford CoreNLP, "TokensRegex Release history." [Online]. Available: https://nlp.stanford.edu/software/tokensregex.html#Usage. [Accessed: 15-May-2017].

[20] R. Valdez Almada, O. M. Rodriguez, C. E. Rose Gómez, and M. de J. Velázquez Mendoza, "Avances de Investigación del Estado de Sonora," *Propuesta de una arquitectura para un sistema generador de perfiles de conocimiento*. Tecnológico Nacional de México, Hermosillo, pp. 265–270, Oct-2016.

[21] M. R. J. Qureshi, "A Comparison of Model View Controller and Model View Presenter," *Sci. Int.*, vol. 25, Aug. 2013.

[22] Yii Software LLC, "Yii Framework." [Online]. Available: http://www.yiiframework.com/. [Accessed: 01-Jan-2017].

[23] S. Meystre and P. J. Haug, "Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation," *J. Biomed. Inform.*, vol. 39, no. 6, pp. 589–599, Dec. 2006.