Journal of The Korea Society of Computer and Information
Vol. 24 No. 7, pp. 19-29, July 2019
www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.07.019

# A Comparative Study between LSI and LDA in Constructing Traceability between Functional and Non-Functional Requirements

Sung-Hoon Byun*, Seok-Won Lee**

## Abstract

Requirements traceability is regarded as one of the important quality attributes in software requirements engineering field. If requirements traceability is guaranteed then we can trace the requirements' life throughout all the phases, from the customers' needs in the early stage of the project to requirements specification, deployment, and maintenance phase. This includes not only tracking the development artifacts that accompany the requirements, but also tracking backwards from the development artifacts to the initial customer requirements associated with them. In this paper, especially, we dealt with the traceability between functional requirements and non-functional requirements. Among many Information Retrieval (IR) techniques, we decided to utilize Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) in our research. Ultimately, we conducted an experiment on constructing traceability by using two techniques and analyzed the experiment results. And then we provided a comparative study between two IR techniques in constructing traceability between functional requirements and non-functional requirements.

▸Keyword: Requirements Traceability, Requirements Engineering, Information Retrieval, Latent Semantic Indexing, Latent Dirichlet Allocation

## I. Introduction

요구사항 추적성(Requirements Traceability)은 개발 초기의 고객의 필요(Needs)로부터 요구사항 문서, 설계, 개발, 배포, 그리고 유지 및 보수 전 과정에 걸쳐 추적할 수 있어야 한다. 이는 요구사항으로부터 수반되는 개발 산출물을 추적하는 것뿐 아니라, 개발 산출물로부터 그와 연관된 최초 고객 요구사항을 추적하는 역방향으로의 추적성도 포함한다[1]. 특히 요구사항 추적성은 실무에서 안전성이 중요시되는 영역 (Safety-Critical Domain)에서 필수적인 속성이다. 그 이유는 안전성 요구사항이 어디에서 기원하였는지를 보여주고 안전성 요구사항의 존재 근거를 제공해주기 때문이다[2]. 요구사항 추적성이 잘 구축된 소프트웨어 시스템 혹은 소프트웨어 프로젝트는 소프트웨어 프로세스 산출물(Software Process Artifacts)간의 연관성 이해와 변경 요청 관리(Change Request Management)를 손쉽게 할 수 있다.

하지만 요구사항 추적성이 소프트웨어 프로젝트의 성공에 주요한 영향을 미치는 요인임에도 불구하고, 요구사항 추적성은 좋은 방법론 및 도구 (Tools)가 부족하고 추가적 비용 및 인력 자원이 요구되기에 자주 무시되었다[3, 4]. Knethen 과 Paech에 따르면, 추적성을 달성하려는 목적의 불분명함도 추적성에 대한 접근법을 제한시키는 요소이다[5].

요구사항 추적성에 관련된 연구들 중에서 특히 요구사항 추적성의 자동화는 굉장히 중요한 부분으로 여겨져 왔다. 요구사항 추적성 연구의 지도적 위치에 있는 Jane Cleland-Huang, Orelna Gotel, Jane Huffman Hayes 의 2014년 연구에서는 여러 가지 요구사항 추적성 연구의 과제들을 제시하였는데, 그 중에서도 궁극적인 과제

• First Author: Sung-Hoon Byun, Corresponding Author: Seok-Won Lee
*Sung-Hoon Byun (byunsunghoon@naver.com), Dept. of Computer engineering, Ajou University
**Seok-Won Lee (leesw@ajou.ac.kr), Dept. of Software and Computer engineering, Ajou University
• Received: 2019. 04. 15, Revised: 2019. 06. 05, Accepted: 2019. 06. 06.
• This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2017R1D1A1B03034279)
• Authors would like to thank Jin-Seoung Oh for his comments and contribution.

(Grand-Challenge)로 편재성 (Ubiquitousness) 제시하였다. 편재성이란 요구사항 추적성은 실무자들이 추적성에 대해 생각할 필요도 없이 저절로 항상 달성되어야 한다는 것을 의미한다. 즉, 요구사항 추적성은 아무런 노력 없이도 (with near zero effort) 달성되고, 유지되어야 한다. 그러기 위해서는 요구사항 추적성과 관련된 업무 (Tasks)가 소프트웨어 공학 프로세스 (Software Engineering Process)에 완전히 정착되어 있어야 한다[6]. 추적성 자동화를 위해 정보검색 (Information Retrieval) 혹은 텍스트마이닝 (Text Mining) 기법을 이용한 요구사항 추적성 접근법이 많이 시도되었다. 정보검색을 활용한 추적성 연구는 추적성 자동화를 위해 꼭 필요한 분야라고 할 수 있다. 그중에서도 잠재 의미 색인 (Latent Semantic Indexing)이 가장 활발히 사용되었다. 그 외 토픽 모델링 (Topic Modeling) 기법인 잠재 디리클레 할당 (Latent Dirichlet Allocation)을 사용한 추적성 연구도 많이 시도되었다. 본 연구에서는 이 두 가지 기법을 이용하여 추적성 구축을 실험하여 유효성을 검증하고, 두 가지 기법에 대한 비교 연구를 진행하였다. 지금까지 잠재 의미 색인 기법과 잠재 디리클레 할당 기법에 대한 비교 연구를 진행한 경우가 Asuncion의 연구[7]와 Williams의 연구[8] 외에는 없다. 더욱이 Williams의 연구는 두 기법의 정확성을 비교하는 연구가 아니기에 연구 가치가 있다고 판단하였다[8].

본 연구에서는 추적성 구축에 대한 범위를 요구사항 명세서 내로 한정하였다. 요구사항 명세서는 대부분 자연어 (Natural Language)로 작성되기에 정보검색 기법을 이용한 추적성 연구에 효과적이기 때문이다. 요구사항 명세서 (Requirements Specification Documents)내에서도 특히 기능 요구사항 (Functional Requirements)과 비기능 요구사항 (Non-Functional Requirements)간의 추적성 구축을 다루었다.

Dan의 연구에 따르면 기능 요구사항과 비기능 요구사항 간의 추적성이 수동으로 이루어질 경우 막대한 비용과 노력이 소요 된다[9]. 또한 Cysneiros의 연구[10]와 Gross의 연구[11]에 따르면, 비기능 요구사항은 시스템 전반위에 영향을 미치기 때문에 완벽하게 추적하기 힘들지만 모든 기능 요구사항들이 비기능 요구사항들을 충족시켰는지를 보장하기 위해서는 이 둘 간의 추적성 구축이 필요하다.

Jane Cleland-Huang 의 연구 에서는 목표 중심 추적성 (Goal-Centric Tracaeability)을 달성하고자 하였는데 소프트골(Softgoal)이라 불리는 상위 개념의 목표로 비기능 요구사항을 모델링하였고, 그 상위 개념의 목표들이 하위 개념의 기능 (Functionality)들로 분해 (Decomposed) 된다[12, 13]. 이 경우도 결국 비기능 요구사항이 분해되어 기능 요구사항이 되는 접근법으로 볼 수 있기 때문에 추적성이 필수적으로 기록되고 보존되어야 한다.

궁극적으로 본 연구에서는 추적성 자동화 연구에 사용되는 잠재 의미 색인 기법과 잠재 디리클레 기법의 기능 요구사항과 비기능 요구사항 간 추적성 구축에 대한 유효성 (정확도)을 실험하고, 두 가지 기법의 비교 연구를 진행하였다.

## II. Related Works

### 1. Requirements Traceability

요구사항 추적성에 대한 정의는 Gotel 과 Finkelstein의 연구가 널리 받아들여지고 있다[1]. 이 정의에 따르면 요구사항 추적성은 요구사항의 기원부터 명세화를 거쳐 이후의 배포와 운용까지 요구사항의 일생을 표현하고 따라갈 수 있는 능력이다. 요구사항 추적성에는 두 가지 기본 타입이 있는데 요구사항 명세화 이전과 관련된 추적성 (Pre-Requirements Specification Traceability i.e. Pre-RS Traceability) 과 요구사항 명세화 이후와 관련된 추적성 (Post-Requirements Specification Traceability i.e. Post-RS Traceability)이다. 요구사항 명세 이전과 관련된 추적성은 이해당사자들의 최초 필요들이 정제되어서 하나의 요구사항이 생성될 때, 이들 간의 관계를 추적할 수 있는 능력이다. 요구사항 명세 이후와 관련된 추적성은 요구사항으로부터 생성되는 소프트웨어 개발 산출물들간의 관계를 추적할 수 있는 능력을 말한다.

모델 기반 개발 (Model-Driven Development)방법론 혹은 모델 기반 공학 (Model-Driven Engineering) 분야를 추적성 연구에 적용한 연구가 활발히 진행되어 왔다. 모델 기반 개발과 모델 기반 공학은 같은 개념으로 봐도 무방하다[14]. 모델 기반 공학은 소프트웨어 개발 단계의 주요 산출물로써 모델을 사용하는 분야이며, 소프트웨어 개발 프로세스를 부분적으로 자동적 모델 변형 (Automatic Model Transformation)을 통해 수행한다. 모델 변형의 과정과 결과를 기록하는 것이 굉장히 중요하고 그 기록들은 추적성의 기반이 된다[14, 15]. 소프트웨어 시스템이 모델의 변형을 통해 개발되기 때문에 어느 한 모델의 변경은 다른 모델에 영향을 미치게 된다. 각 모델 간의 변경 영향력 분석 (Change Impact Analysis)을 위해서는 추적성이 필요하다. 또한 추적성은 소프트웨어 개발 프로세스 내에서의 각 단계의 모델 변형을 문서화하는 (Documentation) 것에서 달성되기 때문에 모델 기반 공학은 추적성 달성에 적합한 소프트웨어 프로세스라고 할 수 있다[14, 15]. 이 모델 기반 공학 접근법 내에서는 소프트웨어 산출물들 (요구사항, 설계 다이어그램, 코드 등)의 모델이 메타모델에 의해서 추출된다. 추적성을 기록하려는 산출물 타입들 (Artifact Types)과 그것들의 관계들 (Relations)을 정의한 모델을 추적성 메타모델 (Traceability Meta-model)이라고 할 수 있다. 이 추적성 메타모델은 추적성 연구 분야에 있어서 많은 연구자들에 의해 연구되어 왔다 [16, 17, 18]. 이 메타모델은 실무자들이 소프트웨어 시스템의 개발을 진행할 때에 추적성을 추출하고 문서화 및 명세화 하는 데에 있어 지침이 될 수 있다. 하지만 Stefan의 연구에 의하면 하나로 표준화된(Standardized) 추적성 메타 모델은 존재하지 않는다[14].

Hayes에 따르면, 정보검색 기법들은 요구사항 분석가들이 직접 수행하는 것보다 훨씬 많은 양의 추적성 연결 (Traceability Links)들을 적은 시간 안에 도출해낼 수 있다[19]. 이러한 IR 기법 중에서 잠재 의미 색인 (Latent Semantic Indexing)이 가장 활발히 이용되었다. 그 외의 경우는 토픽 모델링(Topic Modeling) 기법인 잠재 디리클레 할당 (Latent Dirichlet Allocation)인 경우가 많기 때문에

추적성 연구에 있어서 정보를 활용한 추적성 연구도 시도되었다. 본 논문에 사용된 두 가지 정보검색 기법 LSI과 잠재 디리클레 할당에 대한 자세한 설명은 다음 절에서 다룬다.

## 2. Information Search Method

### 2.1 Latent Semantic Indexing

잠재 의미 색인 (Latent Semantic Indexing)은 잠재 의미 분석 (Latent Semantic Analysis)이라고도 표현한다. LSI은 벡터 공간 모델을 기반으로 하며, 모든 문서 집합인 코퍼스 (Corpus)내의 단어 사용에 잠재적인 구조 (Latent Structure) 가 있다고 가정을 하는 정보검색 기법이다[20]. 정보검색에서 단어 사용의 잠재적 구조 (Latent Structure)가 중요한 이유는 단어들의 다의성 (Polysemy)과 동의성 (Synonymy) 때문이다. 다의성은 lie 와 같이 하나의 단어가 '거짓말하다' 와 '눕다' 의 전혀 다른 여러 가지 뜻을 가지는 특성이다. 동의성은 car 와 automobile 같이 다른 별개의 단어들이 동일한 의미를 가지는 특성이다. 이러한 특성은 정보검색에서 어려움으로 작용한다. LSI은 이러한 문제점을 극복하고자 하는 기법이다[20, 21]. 잠재적인 구조를 파악한다는 것은, 문맥 혹은 맥락이라고 번역되는 글의 컨텍스트 (Context)를 파악한다는 것과 같다. 이를 위하여, LSI은 단어들의 공기 (Co-occurrence)관계를 이용한다. 공기 관계란 글에서 함께 자주 출현되는 단어들이 가지는 관계를 의미한다. 예를 들어 '그곳에 그녀가 살고 있다'라는 문장은 옳은 문장이고 '그곳에 학교가 살고 있다'라는 문장을 틀린 문장이다. 따라서 그녀와 살다는 공기 관계가 있지만, 학교와 살다는 공기 관계가 없다.

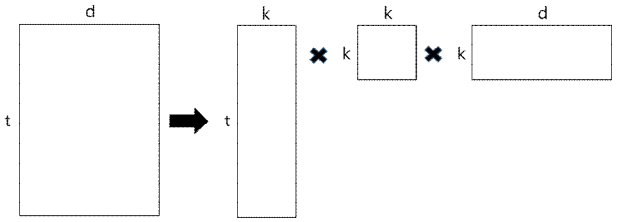


Fig. 1. Singular Value Decomposition of t × d metrix

LSI은 다변량 통계 분석의 하나인 특이값분해 (Singular Value Decomposition)를 이용한다. 특이값 분해가 수행되는 절차는 t 개의 단어와 d 개의 문서가 있다고 할 때, t x d 매트릭스를 적절한 값의 k 를 통해 t x k 와, k x k 그리고, k x d 세 개의 매트릭스의 곱으로 분해하는 과정이라고 할 수 있다. 이를 그림으로 표현하면 위 (Fig. 1)과 같다. 간단히 말해, LSI는 실제 존재하는 t 와 d 의 높은 수치를 k 차원 벡터 공간으로 축소하여 간략화하는 개념이다.

적절한 k 값을 확정하는 것은 매우 중요하다. 실제 텍스트 데이터를 반영할 만큼 충분히 큰 k 값이면서 동시에 중요치 않은 디테일들은 무시할 만큼 작은 k 값을 정해야 한다. 적절한 k 값은 고르는 것은 LSI 분야에서 중요한 토론 대상이다[20, 22]. Bradford의 연구에 의하면 200 에서 500 사이의 k 값이 가장 이상적이다[23]. Yee Whye Teh의 연구에서 이 k값을 자동으로 찾아주는 방법으로 Hierarchical Dirichlet processes (HDP)를 제안하였다[24].

이후에 문서 열을 기준으로 상관분석 (Corrlation Analysis)을 수행하면 문서들 사이의 유사도를 계산할 수 있다. 유사도를 측정할 때, 상관관계의 정도를 나타내는 상관계수 (Correlation Coefficient)로써 코사인 유사도가 일반적으로 사용된다.

LSI은 불용어 목록 (Stop Words List)을 사용한다. 불용어 목록이란 in, between, of, am, are 와 같은 단어처럼 너무 출현 빈도가 높기에 문서 내에서 가치가 없어 단어 집합에서 제외되는 단어들의 집합이다.

### 2.2 Latent Dirichlet Allocation

잠재 디리클레 할당 (Latent Dirichlet Allocation)은 토픽 모델링 (Topic Modeling)이라고도 불린다. 자연어처리 분야에서 LDA는 사전에 주어진 코퍼스에 대하여 코퍼스 내의 각 문서가 어떤 토픽을 가지고 있는지를 추측하는 확률적 모델 (Probabilistic Model)이다[25]. LDA은 2003년에 D. Blei, M. Jordan에 의해 소개되었다.

LDA은 생성적 확률 모델 (Generative Probabilistic Model)이기도 하다. 생성적 확률 모델에서는 문서가 가질 수 있는 토픽들에 대한 확률 분포를 알고 각 토픽에서 나올 수 있는 단어들의 확률 분포를 안다면, 이 확률에 의해 랜덤 프로세스 (Random Process)를 이용해 단어를 하나씩 생성해 나가며 문서를 생성해 낼 수 있다고 가정한다. 이러한 가정 하에 코퍼스가 있으면 이 확률 분포들을 추정하여 문서들의 토픽들을 찾는 방법이 LDA이다. LDA에서의 토픽은 토픽을 구성하는 단어들의 확률 분포라고 할 수 있다.

또한 LDA은 비감독 모델 (Unsupervised Model)이기도 하다. 비감독 모델이란 어느 문서 내의 단어들이 주어졌을 때, 그 단어들이 속할 토픽이 어떤 것들이 있는지 모르는 상태에서 비감독 분류 (Unsupervised Classification)을 하는 것을 의미한다.

LDA이 정보검색 분야에서 주목 받은 이유는 LDA에서는 하나의 문서를 토픽들의 혼합체 (Mixture of Topics)로 간주한다는 점 때문이다. 예를 들어 "박근혜 정부의 의료 민영화 정책, 의사 협회의 파업으로 이어지다."라는 문서가 있다면, 이 문서는 직관적으로 보기에 '의료'라는 토픽이 떠오른다. 하지만, 한편으로는 '정치'라는 토픽도 포함하고 있음을 알 수 있다. 임의적으로 토픽 별로 가중치를 매긴다면 의료 60%, 정치 40%로 매길 수 있을 것이다. 이와 같이 LDA은 모든 문서들이 단 하나의 토픽에 국한되는 것이 아니라, 여러 가지의 토픽들이 가중치를 가지면서 혼합되어 있다고 가정하는 것이 가장 큰 특징이다.

LDA을 도식적 모델로 표현하면 아래 (Fig. 2) 와 같다.

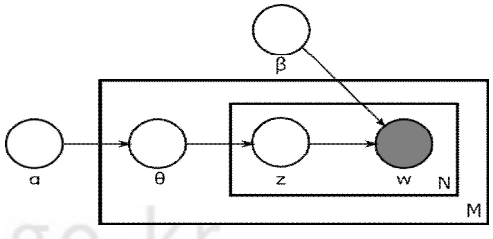


Fig. 2. Schematic model of LDA

이 방법을 통해 우리는 가지고 있는 코퍼스로부터 문서 별 토픽 가중치가 담긴 Θ를 추정할 수 있다. 그렇게 추정된 이 Θ를 이용하여 문서간 유사도를 알아내거나, 문서들을 분류하는 데에 기여할 수 있다. 여기서 D. Blei의 LDA을 개관하는 논문에 따르면, 매개 변수들을 추정하는 방법에는 변분 추론 (Variational Inference)과 깁스 샘플링 (Gibbs Sampling) 이 있다[25]. Mei 연구에서는 도출된 토픽이 무엇을 의미하는지 사람이 추측하기 쉽도록 자동적으로 토픽에 라벨링(labeling)을 해주는 연구를 진행 했다[26].

### 2.3 Research of Traceability using IR

가장 먼저 정보검색을 추적성 연구에 활용한 연구자는 Antoniol이다. Antonio의 2000년 연구에서 소스 코드와 문서들 간의 추적성을 재구축(Reconstruction)하기 위한 정보검색 모델을 제안했다[27].

이어서 Antoniol의 연구와 같은 주제로, A.Marcus의 2003년 연구에서는 LSI을 활용하여 소스 코드와 문서 산출문들 간의 추적성을 연구하였다. 실험을 통하여 LSI을 이용한 소스 코드와 문서 산출물들 간의 추적성 재구축의 가능성을 보여주었다[28].

Jane Hayes의 2003년 연구에서는 추적성 연구 분야에서 정보검색의 중요성을 소개하였다. 또한 Vanilla 알고리즘 등 여러 개의 알고리즘을 적용해 보고 비교하였으며 앞 글자들을 따서 새로운 용어를 만드는 두문자어(Acronyms)나 모호한 요구사항과 같은 문제점들을 발견하였고 추후 더 많은 연구가 필요함을 역설했다[29].

Marco Lormans 와 Arie Van Deursen의 2006년 연구에서는 LSI을 이용하여 요구사항과 테스트 문서 간의 추적성과 요구사항, 그리고 설계 산출물(Design Artufacts)간의 추적성을 구축하는 것에 대한 실험을 했다[22].

Andrea De Lucia et al.의 2009년 연구에서는 LSI을 이용한 연구에서의 명사의 역할과 중요성을 역설하였다[30].

UC Irvine의 ISR (Institute for Software Research)의 Hazeline Asuncion, Richard Taylor가 2010년 진행한 연구는 본 연구와 비슷하게 LSI과 LDA을 추적성 구축에 이용하여 실험을 진행하였다[7]. 이 연구에서는 TRASE라는 LDA과 키워드 기반의 정보검색 기법을 혼합한 툴을 독자적으로 제작하였고, 이 툴을 이용하여 소프트웨어 아키텍처내의 아키텍처적인 요소들(Architectural Components)의 연관성을 자동화한 뒤 이를 시각화 하여 보여 주었다.

Jane Hayes의 2011년 연구에서 연구 결과, 사람이 수동으로 점검(Review)해야 하는 항목들의 수를 줄임으로써 노력이 58% 정도 줄어 효율성이 증대 되었다고 한다[9]. 즉, 기능 요구사항과 비기능 요구사항 간의 추적성 자동화는 실무에서 비용적 이득을 가져다 준다고 할 수 있다.

다음으로 LDA을 추적성 구축에 활용한 연구자는 Rocco Oliveto, Denys Poshyvanyk, Andrea De Lucia가 있다. Oliveto는 LSI 기법이나 Jensen-Shannon 기법은 동등한 성능을 보이며 LDA 기법은 이들보다 정확도는 약간 떨어지지만, 이들이 도출하지 못하는 다른 관점(Dimension)의 정보를 찾아 낼 수 있다고 하였다[31]. 또한 2011년 Gethers의 연구에서는 Jensen-Shannon과 LSI의 기법에 LDA 기법을 혼합해서(Combine) 이용하면 더 정확도를 향상시킬 수 있다고 하였다[32]. 이 연구에서 Gethers은 정보검색 기법들을 혼합하는 방법(Hybrid Method)이 하나의 (Stand-alone) 정보 검색 기법보다 더 나은 결과를 낸다고 하였다.

## III. Open Source Library and Data

### 3.1 Gensim library

본 연구에서는 LSI과 LDA 기법의 사용을 위하여 체코의 연구자 Radim가 2010년에 발표한 Gensim (Generate Similar)이라고 하는 파이썬 언어 기반의 오픈소스 라이브러리를 활용하였다[33]. 이 라이브러리는 LSI과 LDA 알고리즘을 빠르게 익히고 사용할 수 있도록 만들어졌으며, 많은 자연어처리 연구자들에 의해 정보검색 실험의 프로토타이핑을 하는 데에 사용되어 왔다. 윈도우 운영체제에서도 사용이 가능하지만 호환성 측면에 있어서 원활한 사용을 위해 본 연구에서는 페도라 리눅스 운영체제에서 실험하였다. Gensim 라이브러리는 알고리즘 내의 계산을 위해 NumPy 및 SciPy 파이썬 패키지를 이용한다.

### 3.2 GPM Requirement Specification

본 연구에서 사용한 요구사항 명세서는 2000년도부터 2003년도까지 진행된 Global Personal Marketplace(GPM)이라는 미국에서 진행된 프로젝트의 요구사항 명세서이다. 이 요구사항 명세서를 본 연구의 실험 대상 요구사항 명세서로 선정한 이유는 소프트웨어공학 연구 분야에서 권위 있는 미국 카네기 멜론 대학 소프트웨어 공학연구소 (SEI: Software Engineering Institute)의 Donald Firesmith 가 작성한 요구사항 명세서이기 때문에 요구사항 명세서 내용의 질적 측면에서 신뢰도가 높다고 판단했기 때문이다. 또한 일반에게 공개되어 실제로 획득할 수 있는 요구사항 명세서 중에서도 가장 방대한 양의 내용을 포함하고 있고 기능 요구사항의 경우에 굉장히 세세하게 서술되어 있기에 실험에 적합하다고 판단하였다.

실험을 위하여 총 110개의 기능 요구사항 중에서 서술이 불명확한 13개의 요구사항을 생략하여 총 97개의 기능 요구사항을 실험에 사용했다. 또한 총 21개의 비기능 요구사항이 서술되어 있는데 이들 중에서 의미의 중복 및 서술의 모호함을 고려하여 몇몇의 비기능 요구사항을 합치고 생략함으로써 총 9개의 비기능 요구사항을 도출하였다. 하지만 추적성의 경우, 이 요구사항 명세에는 추적성과 관련한 정보가 따로 서술되어 있지는 않았다. 때문에 기능 요구사항과 비기능 요구사항 간의 추적성 매트릭스를 제작하였고, 이를 LSI 및 LDA 실험 결과로 도출된 유사도와 비교하였다. 이 요구사항 명세서를 편의를 위해 Global Personal Marketplace를 앞 철자를 따서 GPM으로 명명했으며 이후 본 논문에서도 GPM이라는 표현의 사용하겠다.

# Ⅳ. Experiment about GPM Requirement Specification

## 4.1 Experiment Environment and Method

GPM요구사항 명세서는 안에 있는 자연어로 된 모든 기능 요구사항과 비기능 요구사항 데이터를 txt 파일 형식으로 수작업으로 변환시켰다. 이 과정에서 유즈케이스 다이어그램과 같은 그림 정보들은 제외하였다. 또한 쌍 따옴표 및 따옴표 같은 정보는 삭제하거나 txt 파일에서 새로 작성하였다. GPM요구사항 명세서는 마이크로소프트 워드 파일로 작성이 되어 있었는데, 워드 파일의 특성상 워드만이 가지고 있는 특수 기호 및 화이트 스페이스 (White Space)정보는 모두 제거 하였다. 그 이유는 Gensim 라이브러리 사용 중에 오류가 나게 되기 때문이다. 이렇게 만든 txt 파일 형식의 기능 요구사항과 비기능 요구사항의 데이터를 이용하여 실험을 진행하였다. 리눅스 상의 터미널에서 Gensim 라이브러리를 임포트(import)하여 파이썬 언어 기반으로 실험이 진행되었다.

먼저 기능 요구사항을 97 개 전체를 하나의 코퍼스 (Corpus)로 간주한다. 그리고 코퍼스 내의 전체 단어 (Terms)를 모두 추출하였다. 모든 단어 중에서 불용어 목록 (Stop Words List)을 제거하였다. 불용어 목록을 제거한 결과, 총 897 개의 개별 단어가 도출되었다. 불용어 목록은 Ranks NL 이라는 독일의 검색 엔진 회사에서 제공하는 불용어 목록을 사용하였다[34]. Gensim 에서는 전체 코퍼스 내에서 단 한 번만 나타나는 단어는 제거하기를 권장하지만, LSI과 LDA 문헌들에서 코퍼스 내에서 한 번만 나타난 단어를 제거하는 것에 대한 내용을 찾아 볼 수가 없어 이들을 단어 집합에 포함 시키는 것이 낫다고 판단하였다. 각 단어들 마다 토큰 id 를 할당하고, 각 단어마다의 출현 횟수 정보와 관련된 통계 정보들을 저장한다. 총 89 개의 토큰 (단어)가 있으며 각 단어 별로 순서대로 총 897 까지의 수로 id 매칭이 된 정보가 나타난다.

```
>>> print(corpus)
MmCorpus(97 documents, 897 features, 4968 non-zero entries)
>>>
>>> for vector in corpus:
...     print(vector)
...
[(19, 2.0), (32, 2.0), (63, 1.0), (210, 4.0), (254, 1.0), (
, (464, 5.0), (466, 1.0), (528, 1.0), (546, 1.0), (552, 1.0
1.0), (870, 1.0)]
[(19, 2.0), (32, 2.0), (63, 1.0), (198, 1.0), (254, 1.0), (
, (457, 4.0), (464, 5.0), (466, 1.0), (528, 1.0), (552, 1.0
1.0), (870, 1.0)]
```

Fig. 3. 897 × 97 Word-Document Matrix

위 (Fig. 3)은 코퍼스 내에 총 97 개의 문서와 897 개의 단어가 있다는 것을 보여준다. 하단은 각 문서 벡터 별로 첫 번째 문서 벡터부터 가지고 있는 단어의 id 와 출현 횟수를 출력한 것을 보여준다.

다음으로 9개 각각의 비기능 요구사항은 상기한 기능 요구사항 벡터 공간에 대한 질의 (Query)가 된다. 즉 하나의 비기능 요구사항을 97 개의 기능 요구사항 문서들에 대하여 유사도 (Similarity) 를 계산하기 위해 질의로써 실험하게 된다.

위의 과정 이후에 LSI과 LDA을 각각 실험하고, 그 결과를 2절의 추적성 매트릭스와 비교하고 그 결고를 분석해보는 실험을 진행하였다.

지금까지 설명한 단계 전의 세부적인 데이터 조작 사항들은 소개하지 않고 생략하였다. 요구사항 명세서의 97 개의 기능 요구사항을 자연어 데이터로 Gensim 에 입력하는 단계, 그리고 그 기능 요구사항에 대한 텍스트로부터 사전 (dictionary)을 만드는 단계, 그리고 사전을 이용하여 97개의 기능 요구사항 자연어 데이터를 897 x 97 단어 – 문서 매트릭스의 벡터 공간으로 만드는 단계가 생략되었다.

## 4.2 Requirements Traceability Matrix

본 절에서는 GPM 요구사항 명세서의 기능 요구사항과 비기능 요구사항 간의 추적성 매트릭스에 대하여 설명하고자 한다. 3절에서 이미 언급하였듯이 GPM 요구사항 명세에는 추적성과 관련한 정보가 따로 서술되어 있지는 않기 때문에 최대한 객관적으로 판단하여 제작하였다.

아래 <Table 1> 은 전체 요구사항 추적성 매트릭스 중 일부

Table 1. Requirements Traceability Matrix

| | Correctness | Extensibility /Scalability | Interoperability | Maintainability | Performance | Reliability/ Robustness | Reusability | Security | Usability |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 3 |
| 2 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 3 |
| 3 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 3 |
| 4 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 3 |
| 5 | 1 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 3 |
| 6 | 2 | 1 | 1 | 1 | 3 | 3 | 1 | 3 | 1 |
| 7 | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 3 | 1 |
| 8 | 3 | 1 | 1 | 1 | 1 | 3 | 1 | 3 | 2 |
| 9 | 3 | 1 | 1 | 1 | 2 | 2 | 1 | 1 | 3 |
| 10 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 |
| 11 | 3 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | 1 |
| 12 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 3 |
| 13 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 3 |
| 14 | 2 | 1 | 2 | 1 | 3 | 2 | 1 | 2 | 3 |

를 발췌한 것이다. <Table 1>의 상단은 왼쪽에서부터 각각 Correctness, Extensibility/Scalability, Interoperability, Maintainability, Performance, Reliability/Robustness, Reusability, Security, Usability 이다.

실제 Global Personal Marketplace 요구사항 명세서에는 더 많은 항목의 비기능 요구사항이 있었지만 거시적으로 보았을 때 비슷한 개념은 한 항목으로 병합하고 애매모호한 것은 삭제했다. 그 결과 비기능 요구사항을 정확성 (correctness), 확장성 (extensibility / scalability), 상호 운용성 (interoperability), 유지 및 보수 용이성 (maintainability), 속도 및 성능 (performance), 내고장성 (reliability / robustness), 재사용성 (reusability), 보안성 (security), 사용자 편의성 (usability)의 총 9 개의 항목으로 분류 할 수 있었다. 기능 요구사항은 1 부터 97 까지 순차적으로 번호가 부여된 기능 요구사항이 아래 방향으로 나타난다. 각 기능 요구사항 별로 9 개의 비기능 요구사항에 1 에서 3 까지 범위의 추적성 점수를 주었다. 3 : 상, 2 : 중, 1 : 하 의 추적성을 의미한다.

세밀하게 점수를 할당함에 있어서 판단 근거가 굉장히 주관적이고 모호하기 때문에 이와 같은 방식을 택하였다. 유지 및 보수의 용이함을 나타내는 특성인 maintainability와 이 시스템이 다른 시스템의 일부로 재사용될 수 있어야 한다는 특성을 나타내는 reusability는 모든 기능 요구사항에 대해 1 의 점수를 할당하였다. 그 이유는 Global Personal Marketplace 요구사항 명세서의 개별의 기능적인 사항들이 GPM 시스템의 유지 및 보수 용이함과 재사용 가능성에 대한 특성에 대한 묘사를 전혀 하지 않았기 때문이다.

### 4.3 Result of Experiment about LSI

본 절에서는 4.2 절에서 도출된 추적성 매트릭스와 LSI의 유사도 도출 실험 결과를 비교한다. 먼저 2 장의 배경지식 부분에서 언급되었던 LSI의 k 값에 결정에 대하여 서술하겠다.

2 장에서 전기하였듯이, LSI의 k 값은 실제 텍스트 데이터를 반영할 만큼 충분히 크면서 중요치 않은 디테일들은 무시할 만큼 작은 값으로 설정해야 한다. 본 연구에서는 임의로 k값을 40으로 할당했다. 그 이유는 본 연구에 사용된 Global Personal Marketplace 요구사항 명세서의 자연어 텍스트 양이 타 연구에 비해 상대적으로 매우 적기 때문이다.

먼저 LSI을 적용하여 유사도를 도출하는 실험 과정에 대하여 설명하겠다. 여기서 설명하는 과정은 다음 4.4 절의 LDA에서도 같은 방식으로 적용된다.

```
>>> tfidf = models.TfidfModel(corpus)
2017-06-07 11:06:31,033 : INFO : collecting document frequencies
2017-06-07 11:06:31,036 : INFO : PROGRESS: processing document #0
2017-06-07 11:06:31,119 : INFO : calculating IDF weights for 97 documents
>>> corpus_tfidf = tfidf[corpus]
>>> lsi = models.LsiModel(corpus_tfidf, id2word=dict, num_topics=40)
2017-06-07 11:08:53,073 : INFO : using serial LSI version on this node
2017-06-07 11:08:53,073 : INFO : updating model with new documents
2017-06-07 11:08:53,153 : INFO : preparing a new chunk of documents
2017-06-07 11:08:53,155 : INFO : using 100 extra samples and 2 power iter
2017-06-07 11:08:53,156 : INFO : 1st phase: constructing (897, 140) acti
2017-06-07 11:08:53,169 : INFO : orthonormalizing (897, 140) action matr
2017-06-07 11:08:53,307 : INFO : 2nd phase: running dense svd on (140, 9
2017-06-07 11:08:53,339 : INFO : computing the final decomposition
2017-06-07 11:08:53,341 : INFO : keeping 40 factors (discarding 16.798%
2017-06-07 11:08:53,355 : INFO : processed documents up to #97
>>> lsiIndex = similarities.MatrixSimilarity(lsi[corpus])
2017-06-07 11:15:32,824 : WARNING : scanning corpus to determine the num
2017-06-07 11:15:32,889 : INFO : creating matrix with 97 documents and 4
```

Fig. 4. Process of LSI

위(Fig. 4)의 제일 첫 번째 줄에서 본 장의 1 절에서 설명한 기능 요구사항의 897 x 97 매트릭스가 담겨 있는 객체 corpus 를 이용해 TF-IDF 모델을 생성한다. 그 이후 객체 corpus 를 TF-IDF 모델을 이용하여 TF-IDF 가중치가 적용된 corpus_tfidf 객체로 변환시킨다. 이 객체와 사전 객체 dict 를 이용, k 값에 40을 할당한 후 LSI 모델 lsi 를 생성한다. 마지막으로 하단에서 corpus 객체를 LSI 모델 lsi 를 이용하여 변환시킨 후, 그 객체를 이용하여 질의의 유사도를 도출하는 데에 사용되는 LSI 유사도 도출 객체 lsiIndex 를 생성한다.

```
>>> cor = "Correctness is the degree to which the system can contain defects and
 still be acceptable to the customer. Latent Defect is the maximum number of all
owable latent defects in released work products. The maximum number of latent bu
gs per unit of software shall not exceed 10 for any release. The maximum number
of tests that can fail when the system is shipped shall not exceed 5. Values of
money shall be correct to the nearest cent. Values of time shall be precise to t
he nearest second."
>>> cor_vec = dict.doc2bow(cor.lower().split())
>>> cor_lsi = lsi[cor_vec]
>>> correctness_lsi = lsiIndex[cor_lsi]
>>> pprint(list(enumerate(correctness_lsi)))
[(0, 0.0083738919),
 (1, 0.012207285),
 (2, 0.011566696),
 (3, 0.010367047),
 (4, 0.013223045),
 (5, 0.0068715056),
 (6, 0.01258986),
 (7, 0.03381905),
 (8, 0.22378793),
```

Fig. 5. Result of LSI Similar calculation of functional requirement of 'Accuracy'

이후 이 lsiIndex 객체를 이용하여 비기능 요구사항을 질의로 입력했을 때, 각 문서 벡터에 대한 유사도를 도출하게 된다. 비기능 요구사항 중 정확성 (correctness)을 질의 (Query)로 넣어 97 개의 기능 요구사항 문서 벡터에 대한 유사도를 도출하는 과정을 위 (Fig. 5)에서 볼 수 있다.

이와 같은 방식으로 9 개의 모든 비기능 요구사항을 질의로 사용하여 LSI 유사도를 도출했다. 이 LSI으로 도출된 유사도를 이용하여, 추적성 매트릭스와 비교해 보는 과정에 대해 설명하겠다. 비교를 위하여 도출된 유사도를 값의 범위에 따라 세 단계로 구분했다. LSI을 이용한 유사도 도출 결과, 각 비기능 요구사항의 기능 요구사항에 대한 유사도의 범위가 대체적으로 0.5를 넘지 않는 것으로 나타났다. 극소수의 몇 개의 유사도는 음수값을 보지만 0에 근접한 음수값이기에 0으로 간주했다. 재사용성 (reusability), 유지 및 보수 용이성 (maintainability), 상호 운용성 (interoperability)의 몇 개의 기능 요구사항에 대한 유사도가 0.6 을 넘는 것을 제외하고는 모든 유사도가 0 에서 0.5 사이의 범위에서 도출되었고 대부분이 0.2 보다 작은 값으로 도출되었다. 유사도의 상대적 크기로 판단했기 때문에, 유사도의 범위를 0 에서 0.1 까지는 '하 (1)', 0.1 에서 0.2 까지는 '중 (2)', 0.2 를 넘는 값은 '상 (3)' 으로 구분하였다. 이 범위에 따라 3 단계화한 유사도 매트릭스는 아래 <Table 2> 과 같다. <Table 2>의 상단은 <Table 1>과 마찬가지로 왼쪽에서부터 차례로 Correctness, Extensibility/Scalability, Interoperability, Maintainability, Performance, Reliability/Robustness, Reusability, Security, Usability 이다.

Table 2. Three Steps Level Similarity of LSI

| | Correctness | Extensibility /Scalability | Interoperability | Maintainability | Performance | Reliability/ Robustness | Reusability | Security | Usability |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| 6 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 3 | 1 |
| 7 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| 8 | 1 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 2 |
| 9 | 3 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 |
| 10 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 2 | 2 |

이렇게 단계화된 LSI 유사도를 2 절의 요구사항 추적성 매트릭스와 비교해 보았다. 9 개 각각의 비기능 요구사항 별로 직접 작성한 추적성 매트릭스와 3 단계화된 유사도 매트릭스 간의 일치하는 비율을 백분율로 아래 <Table 3> 에 표기했다.

Table 3. Accuracy of LSI

| Item | Value |
|---|---|
| Correctness | 34% |
| Extensibility/Scalability | 47% |
| Interoperability | 66% |
| Maintainability | 84% |
| Performance | 26% |
| Reliability/Robustness | 42% |
| Reusability | 85% |
| Security | 51% |
| Usability | 27% |

전체적으로 유지 및 보수 용이성 (maintainability)과 재사용성 (reusability)이 높은 정확도를 보여주었는데 두 비기능 요구사항이 세세한 시스템의 기능적인 사항들과 연관이 되지 않기 때문에 모두 낮은 점수인 '하 (1)' 를 할당하였고, 실제로 유사도 도출 결과 두 비기능 요구사항 모두 97개의 기능 요구사항중의 80% 이상에 대하여 0.1 이하의 낮은 유사도 결과를 도출했기 때문이다. 또한 속도 및 성능 (performance), 사용자 편의성 (usability), 정확성 (correctness)의 경우에는 연관된 기능 요구사항이 많다고 판단하였기에 '상 (3)' 의 점수를 부여한 기능 요구사항이 많았는데, LSI 유사도 결과는 기대와는 다르게 낮은 유사도를 보여준 사항이 많았다. 전체적으로 보았을 때 각 비기능 요구사항의 유사도 도출 결과가 추적성 매트릭스와 비교하여 정확히 일치한다고 보기에는 어려운 정확도 결과를 보여주었다.

## 4.4 Result of Experiment about LDA

이 절에서는 k 값을 45로 정하였다. 45로 정하게 된 배경은 다음과 같다. 적절한 k값을 알기 위하여 k 값에 여러 값을 대입해 보며 추출된 k 개의 토픽들이 얼마나 편향되지 않은 단어 분포를 가지는지 확인해보았다.

그리고 토픽 내 단어들이 편향되지 않게 분포되는 한도 내에서 최대의 k 값으로 정하였다. k 값을 300 으로 입력했을 때에는 토픽 내의 단어들의 구성이 특정 단어로 편향되는 경향을 보였다. 아래 (Fig. 6)에서 k 값에 따른 토픽별 단어의 확률 분포의 일부를 볼 수 있다.

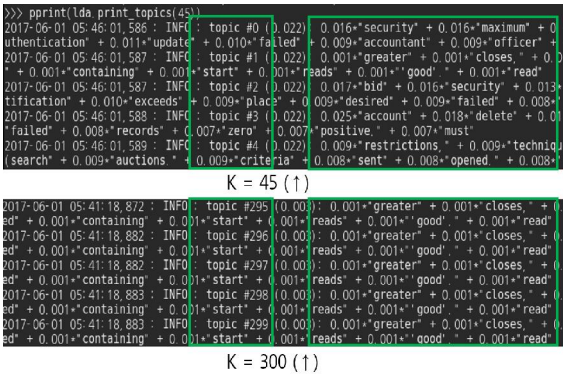


Fig. 6. Word-by-topic distribution depending on K value

또한 k 값이 45일 때는 토픽 #0 이 security, maximum 등으로 확률 분포를 가지고, 토픽 #3 이 account, delete 등을 가진다. 전체적으로 토픽들이 각각 개별의 의미 있는 토픽으로서의 단어 확률 분포를 가지고 있다. 하지만 k 값이 300 일 때에는 토픽 #295 부터 #299 까지 모두 greater, closes 등 같은 단어 분포를 가지고 있다. 또한 k 에 불필요하게 큰 값을 할당했을 경우에는 토픽의 중복이 일어나는 것을 확인했다. 토픽이 고르게 나타나는 k 값을 큰 값에서부터 낮추어 가면서 찾은 값이 40에서 45정도였으며, 최종적으로 45로 확정했다. 그 다음 9개의 모든 비기능 요구사항을 질의로 사용하여 LDA 유사도를 실험을 통하여 도출하였다.

LSI과 달리, LDA은 연관이 없다고 판단되는 기능 요구사항 문서에 대해 유사도 값 0을 바로 도출해내는 것을 확인할 수 있다. 또한 LDA는 LSI에 비하여 상대적으로 고르게 0에서 0.9 근방까지 유사도가 분포되었다.

하지만 LDA도 LSI의 경우와 비슷하게 0.5 이하의 낮은 값에 많이 분포되었다. LSI와 마찬가지로 4.2 절의 추적성 매트릭스와 LDA 기법을 사용하여 도출된 유사도를 비교하기 위하여 도출된 유사도를 값의 범위에 따라 세 단계로 구분하였다. LSI에는 없는 0 값이 많이 도출되었기 때문에 0 자체를 '하 (1)' 단계로 정하였다. 0에서 0.4까지는 '중 (2)', 0.4 보다 큰 값은 '상

Table 4. Three Steps Level Similarity of LDA

| | Correctness | Extensibility /Scalability | Interoperability | Maintainability | Performance | Reliability/ Robustness | Reusability | Security | Usability |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 3 | 2 | 2 | 1 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 | 3 | 2 | 1 | 2 | 3 |
| 4 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 2 |
| 7 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 |
| 8 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 |
| 9 | 3 | 1 | 1 | 1 | 2 | 3 | 1 | 1 | 1 |
| 10 | 2 | 1 | 1 | 1 | 2 | 2 | 1 | 2 | 3 |

(3)'으로 구분하였다. 이에 따라 3단계화된 유사도 매트릭스는 위 <Table4>와 같다.

이를 이용하여 4.2 절의 요구사항 추적성 매트릭스와 단계화된 LDA 유사도를 비교해 보았다. LSI과 마찬가지로 9개 각각의 비기능 요구사항별로 작성한 추적성 매트릭스와 3 단계화된 유사도 매트릭스 간의 일치하는 비율을 백분율로 아래 <Table 5> 에 표기하였다. LSI과 마찬가지로 유지 및 보수 용이성 (maintainability)과 재사용성 (reusability)이 높은 정확도를 보였다. 특히 재사용성은 LSI과 1% 차이가 나는 거의 똑같은 정확도를 보여주었다. 속도 및 성능 (performance)을 제외한 모든 또한 비기능 요구사항이 LSI보다 낮은 정확도를 보여주었다. 그 결과 LSI 기법과 비슷하게 각 비기능 요구사항의 유사도 도출 결과가 추적성 매트릭스와 비교하여 정확히 일치한다고 보기에는 어려운 정확도 결과를 보여 주었다.

Table 5. Accuracy of LDA

| Item | Value |
|---|---|
| Correctness | 27% |
| Extensibility/Scalability | 38% |
| Interoperability | 50% |
| Maintainability | 77% |
| Performance | 43% |
| Reliability/Robustness | 50% |
| Reusability | 86% |
| Security | 40% |
| Usability | 19% |

# Ⅴ. Discussion

## 5.1 Analysis and Evaluation

실험 결과에 대한 평가 및 분석해 보았다. 먼저 각 비기능 요구사항별로 정확도의 상대적 차이가 보여졌다. 특히 비기능 요구사항 중 하나인 재사용성(reusability)이 가장 높은 정확도를 보여 주었다. LSI 기법은 85%, LDA 기법은 86% 의 정확도를 각각 보였다. 이때 재사용성은 GPM 시스템의 일부분 혹은 전체가 다른 시스템 (혹은 더 큰 범위의 시스템)에서 다시 사용되기에 용이함을 의미한다.

또한 유지 및 보수 용이성 (maintainability)이 두 번째로 높은 정확도를 보여 주었다. LSI 기법은 84%, LDA 기법은 77%의 정확도를 각각 보였다.

이 두 가지 비기능 요구사항 재사용성 및 유지 및 보수 용이성은 다른 비기능 요구사항에 비해 상대적으로 훨씬 높은 정확도를 보였다. 첫 번째 이유는 두 비기능 요구사항 모두 성질적으로 세세한 시스템의 기능적인 사항들과 연관이 되지 않기 때문이다. 실제로 GPM 요구사항 명세서의 기능 요구사항들을 읽어 보면 이 시스템이 재사용되거나 고쳐지기 쉽다는 성격의 서술은 전혀 나타나지 않는다. 두 번째 이유는 제작한 추적성 매트릭스에서 낮은 점수를 부여했고, LSI와 LDA 두 기법 모두 전혀 연관성이 없는 자연어 데이터에서는 유사도가 극단적으로 낮게 도출되는 경향을 보이기 때문이다. 이러한 이유들로 높은 정확도가 나타난 것으로 보인다.

정확성 (correctness)과 사용자 편의성 (usability)과 같은 비기능 요구사항은 두 기법 모두 전체적으로 낮은 정확도를 보였다.

LSI 기법은 정확성 : 34%, 사용자 편의성 : 27% 를 보였고, LDA 기법은 정확성 : 27%, 사용자 편의성 : 19% 를 보였다. 상호 운용성 (interoperability)과 보안성 (security)의 경우에는 두 기법 모두 중간 정도의 정확도를 보였다. LSI 기법은 상호 운용성 : 66%, 보안성 : 51% 를 보였고, LDA 기법은 상호 운용성 : 50%, 보안성 : 40% 를 보였다.

종합하여 보면 LSI 기법과 LDA 기법이 서로 다른 정확도 수치를 보였지만 정확도가 도출되는 데에 있어서 비슷한 경향을 보였다. 즉 두 기법이 상대적으로 높은 정확도를 보이는 비기능 요구사항과 상대적으로 낮은 정확도를 보이는 비기능 요구사항의 경향성은 동일했다는 것이다. 이를 통해 판단을 내릴 수 있는 것은 LSI 기법과 LDA 기법이 보여주는 텍스트 데이터 간의 유사도를 도출하는 능력은 비슷하지만, 그 결과가 실제로 사람의 판단과 동일하다고 확신을 할 수는 없다는 것이다. 본 연구 실험 결과에서 또한 LDA 기법이 정확도가 낮았기에 향후 LSI 기법과 LDA 기법의 결과를 복합적으로 적용하여 유사도를 도출하는 연구가 필요하다고 판단할 수 있었다.

## 5.2 Limitations

4장의 정확도 결과에서 LSI 기법과 LDA 기법 모두 2 개를 제외한 비기능 요구사항의 정확도가 60%를 넘지 못하는 정확

도를 보여 주었다. 이와 같이 기대했던 것보다 낮은 정확도의 결과가 나온 원인을 다음과 같은 4가지로 분석해 보았다.

첫째, 요구사항 명세서 하나만으로는 텍스트의 절대량이 부족했다. 일반적으로 텍스트마이닝 혹은 정보검색 기법을 적용하여 연구를 진행할 때는 방대한 양의 코퍼스를 학습 데이터로 사용하는 경우가 많다. 비록 Global Personal Marketplace 요구사항 명세서가 구할 수 있는 요구사항 명세서 중 가장 큰 규모의 요구사항 명세서를 선택한 것이지만 그럼에도 불구하고 정확한 유사도 결과를 도출하기에는 부족한 텍스트 절대량이라고 보여진다. LSI의 경우, 부족한 절대 텍스트량 때문에 유사도 분포가 대부분이 0.5를 넘지 못하고 낮은 범위에서 나온 것으로 판단된다. LSI은 공기성 (co-occurrence)을 이용하는데, 절대적인 텍스트의 양이 부족하면 공기 정보가 부족하게 된다. 때문에 특정 비기능 요구사항을 질의로 넣었을 때의 비기능 요구사항과 기능 요구사항 간의 유사도를 찾을 만한 공기 정보가 부족하게 되고, 낮은 유사도 값이 도출되는 것으로 보여진다.

둘째로는, Gensim 라이브러리에서 단어의 전처리를 하는 기능이 완벽하지 못하다는 점이다. 정보검색 분야에서 '어간 추출 및 분류' 라고 하는 stemming and lemmatization 의 기능을 제공하지 못한다. 포터 알고리즘을 이용하여 접두사 (Prefix) 및 접미사 (Suffix)를 제거하는 기능은 전혀 제공하지 못 한다. 본 저자가 Global Personal Marketplace 워드 파일 원문의 불필요한 문장부호는 모두 수작업으로 제거하였지만 필수적으로 들어가야 하는 따옴표 (Apostrophe), 괄호 (Parenthesis, Bracket), 쉼표 (Comma), 마침표 (Period) 등이 단어와 공백이 없이 붙어 있을 경우에 이 단어들을 모두 그대로 인식해 버린다.

셋째로는 제작한 추적성 매트릭스 자체의 객관성 결여가 낮은 정확도를 유발했을 가능성이 있다. Global Personal Marketplace 요구사항 명세서에 기능 요구사항과 비기능 요구사항 간의 추적성에 대한 정보는 제공되지 않았다. 그렇기 때문에 요구사항 매트릭스를 스스로 제작하였고 최대한 합리적으로 제작하려고 노력하였지만 주관성이 포함되는 것은 불가피하다.

마지막으로 LSI 기법과 LDA 기법의 유사도 결과에 대하여 범위를 설정하고 범위 내의 유사도를 하나의 점수로 간략화하는 방식으로 실험을 진행한 점이 정확도에 부정적 영향을 끼쳤을 가능성이 있다. 추적성 매트릭스 작성 시 얼마나 추적성이 높은지에 대한 판단이 모호할 수 있기 때문에 추적성 매트릭스는 3단계의 점수로 작성이 되었다. 이 추적성 매트릭스와의 비교를 위한 실험 결과 유사도 또한 범위에 따라 간략화 하여 3단계로 나누었다. 유사도를 구간 별로 나누어 동일하게 간주함으로써 정교한 수치의 비교가 이루어지지 못 하였다.

또한 앞서 언급한 Asuncion의 연구의 경우 본 실험과 유사한 방식으로 실험이 진행 되었으나 Asuncion의 연구에서는 LSI이 비기능 추적성에서 더 높은 정확도를 보여주어 본 연구와는 반대의 결과를 보였다[7]. 하지만 이는 Asuncion의 연구에서 아키텍처 요소들(Architectural Components) 간의 추적성 구축에 관해 다루었기 때문에 본 연구와의 직접적인 비교는 무리가 있으며, 텍스트 마이닝 실험의 성격상 실험환경(Environmental set)에 따라 결과가 달라지는 점도 고려해야 한다.

# Ⅵ. Conclusion

## 6.1 Contribution

지난 20년간 많은 연구자들이 추적성 자동화를 위해 다양한 시도를 해왔다. 그 중에서도 정보검색 기법인 LSI와 LDA를 이용한 추적성 구축에 대한 다양한 실험적 연구가 진행되었다. LDA을 이용한 추적성 연구를 진행한 연구자는 UC Irvine의 연구[7] 와 이탈리아 University of Salerno의 연구 [31,32]가 있다. UC irvine 연구[7]에서는 LDA이 더 정확하다고 하였으며 Yee Whye The와 Mei의 연구에서는 LDA이 더 정확하지 못 하다고 하였다[24, 26]. 그 외의 Dan Fort[9]와 De Lucia[27] 그리고 Hayes[29]이 잠재 LSI 기법을 이용하여 추적성 연구를 진행하였다. 결론적으로 아직까지는 LSI 기법이 추적성 연구에 더 활발하다고 할 수 있다.

본 연구에서는 실험을 통해 정보검색의 추적성 구축의 정확도를 실험하여 이러한 두 가지 기법의 비교해 보고 이러한 경향 속에서 어떤 기법이 더 우위에 있는지를 확인하고자 하였다.

그 결과 본 연구에서는 LSI 기법이 LDA기법의 정확성보다 높다는 결과를 얻을 수 있었다. 하지만 종합적으로 보았을 때 두 기법의 경향성 자체는 동일했다. 또한 GPM 요구사항 명세서를 읽어 보았을 때 관련성을 찾아 볼 수 없는 재사용성과 유지 및 보수 용이성이 본 연구 결과에서 높은 정확도를 보인 것으로 통해 LSI기법과 LDA기법을 통하여 얻은 결과가 실제로 사람의 판단과 동일하다고 보기 힘들다는 결론을 얻을 수가 있었다. 보다 정확한 비교를 위하여 LDA를 이용한 추적성 구축에 관한 실증적인 연구가 필요하다는 것 또한 알 수 있었다.

## 6.2 Future Work

정보검색 기법을 이용하여 추적성을 구축하는 데 있어서 자동화 연구는 매우 활발히 이루어지고 있는 주제이다. 추적성 구축에 있어서, 정보검색 기법이 얼마나 효과적인지, 실제로 실무에서 문제없이 쓰일 수 있는 수준인지를 검증하는 연구를 계속 이어나가고자 한다. 정보검색을 이용한 추적성 구축을 위해 가장 효과적인 실험환경이 무엇인지를 알아내고자 하는 연구 또한 가능하다고 판단된다. 실험 환경에는 '적절한 요구사항 명세서의 텍스트 양', '정보검색 기법에 대한 입력 데이터 (Input Data)'로써 이용 가능한 소프트웨어 산출물 형식 (테스팅 문서, 소스 코드 등)이 있을 수 있다.

더 나아가 정보검색 기법을 활용한 추적성 구축 전문 도구 (Tools) 을 구현하는 연구도 쉽지 않은 시도이지만, 그 가치가 매우 큰 연구가 될 것이다. 마지막으로 LSI 기법과 LDA 기법을 혼용하여 추적성 연구에 활용하여 정확도를 높이는 연구 또한 가치가 있을 것이다.

## REFERENCES


[1] Orelena C. Z. Gotel, Anthony C. W. Finkelstein, "*An analysis of the requirements traceability problem*", Proceedings of the First International Conference on Requirements Engineering, Colorado Springs, CO, USA, 1994.

[2] Vikash Katta, Christian Raspotnig, Peter Karpati, Tor Stalhane, "*Requirements management in a combined process for safety and security assessments*", Eighth International Conference on Availability, Reliability and Security (ARES), p.780-786, Regensburg, Germany September 2013.

[3] Vikash Katta, Tor Stalhane, "A conceptual model of traceability for safety systems", Electronic Proc. 2nd Complex Systems Design & Management Conference (CSD&M 2011), Poster-Session Paper, Paris, France, 7-9 December 2011.

[4] Kannenberg, Andrew, and Hossein Saiedian. "Why software requirements traceability remains a challenge." CrossTalk. The Journal of Defense Software Engineering 22.5 : 14-19, 2009

[5] Knethen AV, Paech B, "A survey on tracing approaches in practice and research", Research Report, ESE-Report, 095.01/E, Fraunhofer IESE, Kaiserslautern, 2002.

[6] Jane Cleland-Huang, Orlena C. Z. Gotel, Jane Huffman Hayes, et al, "*Software traceability: trends and future directions*" Proceedings of the on Future of Software Engineering, p. 55-69, Hyderabad, India , 2014.

[7] Asuncion, Hazeline U., Arthur U. Asuncion, and Richard N. Taylor. "*Software traceability with topic modeling.*" Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. p.95, Cape Town, South Africa, 2010.

[8] Thefor P.Williams, John F.Betak. "A Comparison of LSA and LDA for the Analysis of Railroad Accident text." Procedia Computer science, Vol 130, p. 98-102, 2018

[9] Dan Port, Jane Huffman Hayes, LiGuo Huang. "*Text mining support for software requirements: Traceability assurance.*" System Sciences (HICSS), 2011 44th Hawaii International Conference on. IEEE, p.1-11, Kauai, HI, USA, 2011.

[10] Cysneiros, Luiz Marcio, and Julio Cesar Sampaio do Prado Leite. "Nonfunctional requirements: From elicitation to conceptual models." IEEE Transactions on Software Engineering Vol 30, Issue5, p.328-350, 04 May 2004

[11] Gross, Daniel, and Eric Yu. "From non-functional requirements to design through patterns.", Requirements Engineering, Vol 6, Issue 1, p.18-36, 2001

[12] Jane Cleland-Huang, et al. "*Goal-centric traceability for managing non-functional requirements.*" Proceedings of the 27th international conference on Software engineering. Saint Louis, MO, USA, 2005.

[13] Jane Cleland-Huang, "*Toward improved traceability of non-functional requirements.*" TEFSE '05 Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering. p.14-19, Long Beach, California, 2005.

[14] Stefan Winkler, Jens von Pilgrim, "A survey of traceability in requirements engineering and model-driven development", Journal Software and Systems Modeling (SoSyM) archive, Volume 9, Issue 4, Pages 529-565, September 2010

[15] Ismenia Galvao, Arda Goknil, "*Survey of traceability approaches in model-driven engineering*", 11th IEEE International Enterprise Distributed Object Computing Conference, Annapolis, MD, USA, 2007

[16] Stale Walderhaug, Ulrik Johansen, Erlend Stav, Jan Aagedal, "Towards a generic solution for traceability in MDD", ECMDA Traceability Workshop (ECMDA-TW), p. 41-50, 2006.

[17] Pedro Sanchez, Diego Alonso, Francisca Rosique, Barbara Alvarez, Juan A. Pastor, "Introducing safety requirements traceability support in model-driven development of robotic applications", IEEE Transactions on Computers, Vol 60, Issue 8, Aug. 2010.

[18] Kassab, Mohamad, Olga Ormandjieva, and Maya Daneva. "*A traceability metamodel for change management of non-functional requirements.*" Software Engineering Research, Management and Applications, Prague, Czech Republic, 2008

[19] Hayes, Jane Huffman, Alex Dekhtyar, and James Osborne. "*Improving requirements tracing via information retrieval.*" Requirements Engineering Conference, Monterey Bay, CA, USA, 2003

[20] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. "Indexing by latent semantic analysis", journal of the American society for information science, Vol 41, Issue 6, 391, September 1990

[21] Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. "An introduction to latent semantic analysis.", Discourse processes, Vol 25, Issue 2-3 p. 259-284, 1998

[22] Lormans, Marco, and Arie Van Deursen. "*Can LSI help reconstructing requirements traceability in design and test?*" Software Maintenance and Reengineering, Bari, Italy, 2006

[23] Bradford, Roger B. "*An empirical study of required dimensionality for large-scale latent semantic indexing*

*applications*." Proceedings of the 17th ACM conference on Information and knowledge management. p. 153-162, Napa Valley, California, USA, 2008

[24] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal and David M. Blei, "Hierarchical Dirichlet Processes", Journal of the American Statistical Association, Vol 101, No.476, p.1566-1581, 2006

[25] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." Journal of Machine Learning Research 3, p.993-1022, Jan 2003

[26] Mei, Qiaozhu, Xuehua Shen, and ChengXiang Zhai. "*Automatic labeling of multinomial topic models*." Proceedings of the 13th ACM SIGKDD international conference onn Kowledge discovery and data mining, p.490-499, San Jose, California, USA, 2007

[27] De Lucia, Antoniol. "*Information retrieval models for recovering traceability links between code and documentation*." Proceedings 2000 International Conference on Software Maintenance, San Jose, CA, USA, 2000

[28] Marcus, Andrian, and Jonathan I. Maletic. "*Recovering documentation-to-source-code traceability links using latent semantic indexing*." Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, 2003

[29] Hayes, Jane Huffman, Alex Dekhtyar, and James Osborne. "*Improving requirements tracing via information retrieval*.", Proceedings. 11th International Requirements Engineering Conference, p.138-147, Monterey Bay, CA, USA, 2003.

[30] Giovanni Capobianco, Andrea De Lucia, et al. "*On the role of the nouns in IR-based traceability recovery*." , 2009 IEEE 17th International Conference on Program Comprehension, p. 148-157, Vancouver, BC, Canada, 2009.

[31] Oliveto, Rocco, et al. "*On the equivalence of information retrieval methods for automated traceability link recovery*.", 2010 IEEE 18th International Conference on Program Comprehension, p. 68-71, Braga, Minho, Portugal, 2010.

[32] Gethers, Malcom, et al. "*On integrating orthogonal information retrieval methods to improve traceability recovery*.", 2011 27th IEEE International Conference on Software Maintenance(ICSM), p. 133-142, , Williamsburg, VI, USA, 2011.

[33] Rehurek, Radim, and Petr Sojka. "Software framework for topic modelling with large corpora." In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, At Malta, May 2010.

[34] Stopword Lists, *www.ranks.nl/stopwords*

## Authors

Sung-hoon Byun received BS in Computer Science from Gachon University at Seongnam in 2015 and MS in Software Engineering from Ajou university in 2017. His area of interest is Software Engineering, Project Management and Software Process Innovation

Seok-Won Lee received the BS in Computer Science from Dongkuk University at Seoul, Republic of Korea. MS in Computer Science from University of Pittsburgh at Pittsburgh, Pennsylvania, and PhD in Information Technology from George Mason University at Fairfax, Virginia, U.S.A. He is currently a Full Professor in the Department of Software and Compute Engineering at Ajou University, Republic of Korea since 2012. His areas of specialization include software engineering with specific expertise in ontological requirement engineering and domain modeling, and knowledge engineering with specific expertise in knowledge acquisition, machine learning and knowledge-based systems.