PlaynView-DistinctWordFinder

"두 문서 집합에서 두드러지게 나타나는 단어를 찾아라"

프로그램 개요: K팝과 J팝의 과거 10년(2008~2017년)의 연간 톱100 팝음악 가사에서 두 나라의 차별되는 특징 단어가 무엇인지를 발견하는 프로그램.

프로그램 개발자: 조희련(heeryon@gmail.com), 조양진

본 문서 작성일: 2018년 11월 15일

□ 우리가 만들었어요

<사진설명:"건강한 신체에 건강한 결과물이 깃든다." 2018년9월8일(토) 남한산성에서. 왼쪽: 조희련, 오른쪽: 조양진>

K-pop과 J-pop 가사 속에서 차별되는 단어를 추출하는 PlaynView-DistinctWordFinder는 데이터 처리 과정에서 한-일 팝음악 가사를 어느 한쪽의 언어로 통일해야 합니다. 당초에는 네이버 일-한 사전을 활용하여 일본어 단어를 한국어 단어로 자동변환 하는 방법을 고려했으나, 자동변환이 생각만큼 쉽지 않아 결국 한-일 통번역 전문가(팀원 조양진)를 영입하여 일본어-한국어 팝 가사 단어 매핑 사전(J-pop/K-pop lyrics word alignment dictionary)을 구축하여 PlaynView-DistinctWordFinder를 함께 구현했습니다.

1. 개발배경 및 목적

세계 팝음악의 지형이 바뀌고 있습니다. 지금까지 서양에서 전세계로 유통되던 팝음악(예: 비틀즈, 마이클 잭슨 등)이 K팝의 등장으로 한국에서 세계로 유통되기(예: 방탄소년단) 시작했습니다. **세계의 팝음악 제작자들은 국내외로 더 많은 경쟁과 기회에 노출**되고 있습니다.

PlaynView-DistinctWordFinder는 음악제작자 중에서도 **작사가들에게 도움을 주고자 하는** 프로그램입니다. 구체적인 예시로 이번 개발에서는 **K팝과 J팝을 구분 짓는 차별 단어(distinct word)를 추출**하는 프로그램을 구현했습니다. 이 프로그램은 **한국과 일본의 팝음악 작사가들이 상대 국가에 음악을 출시할 때, 해당 국민들이 어떤 단어들에 익숙해 있는지를, 기존 히트 팝송들에 내재된 차별 단어들을 파악함으로써, 현지 소비자들의 감성을 한층 더 자극할 수 있는 가사를 작사할 수 있도록 도와주는 것을 목적**으로 하고 있습니다.

2. 개발환경 및 개발언어

운영체제: Ubuntu 16.04, **소스언어:** Python 3.5, **통합개발환경:** PyCharm, **Python 라이브러리/모듈:** pandas 0.23.3, beautifulsoup4 4.4.1, requests 2.9.1, numpy 1.14.5, scikit-learn 0.19.2, mecab-python3 0.7, konlpy 0.5.1, genism 3.5.0, matplotlib 2.2.2, tensorly 0.4.2

3. 시스템 구성 및 아키텍처

PlaynView-DistinctWordFinder는 **데이터 수집부**(crawl_data)와 **차별단어 추출부**(find_distinct_words)로 나뉩니다.

- **데이터 수집부**(crawl_data): 한-일 팝음악 랭킹 사이트인 멜론(한국)과 오리콘(일본)에서 10년치 연간 톱100 팝음악을 HTTP GET request로 수집했습니다. J팝의 경우 연간 히트곡 순위를 오리콘에서 먼저 파악한 후, 개개 가사는 가사를 제공하는 사이트들에서 수집했습니다.

- **차별단어 추출부**(find_distinct_words): 한-일 팝송 가사 텍스트(text data)를 입력으로 한 후, ① 전처리(tokenization & dictionary index word filtering), ②단어 임베딩 학습(word2vec), ③텐서 분해(CP decomposition), ④단어 정렬(n-words sorting based on mode-1 vector values), ⑤클러스터링 성능 비교(k-means clustering), ⑥K팝/J팝 차별단어 목록 출력의 순서로 데이터 처리를 진행하도록 구현했습니다. 이번 개발에서는 단어의 분석 단위를 명사(K팝의 경우, 일반명사와 고유명사), 동사, 형용사의 세 가지 품사로 결정했습니다.

차별단어 추출부에는 한-일 통번역 전문가가 구축한 J팝/K팝 가사 단어 매핑 사전이 탑재되어 있는데, 이를 활용하여 두 나라의 팝 가사 텍스트를 모두 한국어로 변환하여 차별단어 추출 절차를 진행했습니다. 전체적인 시스템 흐름도를 아래에 제시합니다. (다음 페이지 참조)

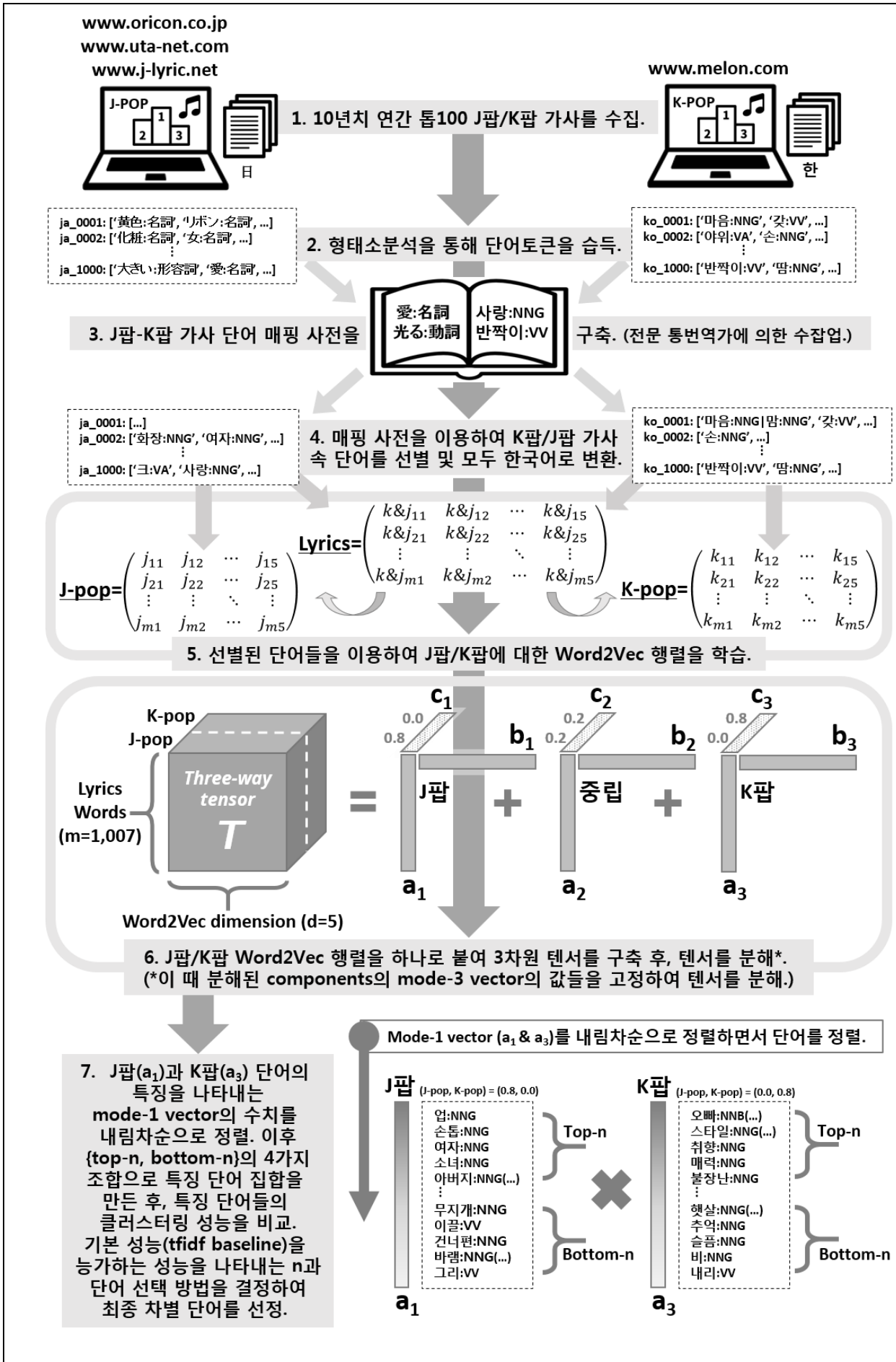

그림 1. PlaynView-DistinctWordFinder 시스템 흐름도

4. 프로그램 주요기능

PlaynView-DistinctWordFinder는 텐서 분해(tensor decomposition)를 계산하는 'tensorly'라는 Python 라이브러리를 수정(modify)하여 차별 단어를 추출하고 있는데, 바로 이 부분이 주요기능입니다.
지금까지 특이값 분해*를 이용하여 문서 벡터의 차원을 축소하여 문서 분석(예: 자동분류, 클러스터링 등)을 진행한 연구개발 결과들은 많았지만, 3차원 이상의 다차원 텐서를 분해하는 방식으로 텍스트를 분석한 연구개발 결과는 아직 많지 않습니다.
인터넷의 등장으로 다양한 텍스트 데이터가 축적되면서, 3차원 이상의 다차원에서(문서, 단어의 변수/축 이외에 시간, 성별, 지역, 국가 등 다양한 변수/축을 고려하여) 텍스트를 분석할 필요성이 대두되면서, 텐서 분해를 활용한 문서 분석 연구개발이 시작되고 있습니다.
PlaynView-DistinctWordFinder는 텐서 분해 중에서도 Canonical Polyadic (CP) Decomposition**이라 불리는 분해 방법을 채택하고 있는데, 분해 결과의 mode-3 vector의 국가별(일본J팝 vs. 한국K팝) 변수의 수치를 고정함으로써, 국가별 차별 단어를 추출할 수 있도록 하였습니다.

*Singular Value Decomposition: SVD; 자연어처리 분야에서는 Latent Semantic Indexing: LSI 또는 Latent Semantic Analysis: LSA로 불림.
** CANDECOMP/PARAFAC decomposition으로도 불림.

5. 기대효과 및 활용분야

본 개발에서는 구체적인 요구사항(use case)으로 J팝/K팝 차별단어 추출을 제시했습니다. 본 개발의 주요기능인 mode-3 vector 수치 고정(fixed) 텐서 분해를 활용할 경우, 다음과 같은 문서 분석을 진행할 수 있습니다.

뉴스 댓글의 남성/여성 시각 차이 분석

화제가 되고 있는 뉴스 사안에 대한 댓글에서, 남녀가 차별적으로 사용하는 단어를 분석함으로써 해당 사안에 대한 남녀의 시각 차이를 분석할 수 있습니다. (Mode-3 vector: {male, female})

두 문서 군의 시간에 따른 전/후 문서의 차이 분석

Lifelog 등을 통해 축적되는 개인이 작성한 문장들(블로그글, 카톡글 등)을 일정 시점을 기준으로 과거와 현재로 나눠, 개인의 글의 변화를 탐지함으로써 정신건강 관리 및 자살방지 등에 활용할 수 있습니다. (Mode-3 vector: {past, present})
또 가요 가사 분석에 적용할 경우, 7080 가요 가사와 2000년대 이후의 가요 가사의 차이를 차별 단어 추출을 통해 분석할 수 있습니다. (Mode-3 vector: {7080, 2000이후})
문학 분야에서는 동일작가의 작풍(writing style)이 작가의 평생에 걸쳐 시간이 지남에 따라 어떻게 달라졌는지를 데이터에 기반하여 분석할 수 있습니다. (Mode-3 vector: {과거작품, 이후작품})

지역별 맛집 경향 분석

가로수길로 가야할지, 경리단길로 가야할지, 홍대 근처로 가야할지 고민일 때, mode-3 vector에 지역을 삽입하여 맛집평을 분석함으로써, 차별적으로 쓰이는 단어들의 분석을 통해, 각 지역의 특징을 파악할 수 있습니다. (Mode-3 vector: {가로수길, 경리단길})

참고로 위에서 제시한 활용분야들은 모두 단일 언어(한국어)의 문서 군을 다루기 때문에, 다국어 단어 매핑 사전(dictionary)의 구축과 단일 언어로의 변환이 불필요합니다(생략됩니다).

6. 공개SW 라이선스

PlaynView-DistinctWordFinder의 주요기능인 텐서 분해 부분은 'tensorly'라는 Python 라이브러리를 수정(modify)하는 것으로 구현하고 있습니다(http://tensorly.org/stable/home.html).
Tensorly가 BSD-3-clause 라이선스를 가지고 있어, 이를 계승하는 형식으로 본 공개SW PlaynView-DistinctWordFinder도 BSD-3-clause로 라이선스를 설정했습니다.

7. 클러스터링을 통한 n개의 단어 설정 및 정렬 방법 결정

텐서 분해 결과에서 Mode-1 vector를 추출하여 이를 내림차순으로 정렬한 후, 아래의 네 가지 경우의 단어를 선택하여(n은 100개 단위로 늘림), 특징 단어로 설정하여 클러스터링을 실시합니다.

- J-pop 상위 n개 단어와 K-pop 상위 n개 단어 (그림 2. jako_both_top)
- J-pop 상위 n개 단어와 K-pop 하위 n개 단어 (그림 2. ja_top_ko_bottom)
- J-pop 하위 n개 단어와 K-pop 상위 n개 단어 (그림 2. ja_bottom_ko_top)
- J-pop 하위 n개 단어와 K-pop 하위 n개 단어 (그림 2. jako_both_bottom)

그리고 아래의 그림 2와 같은 클러스터링 성능 그래프를 작성한 뒤(성능 평가 척도로 Adjusted Rand Index를 사용), **사용자가 본인이 보기에 괜찮은 성능을 선택합니다**. 소스코드에서는 사용자가 n=300 (ja와 ko가 더해져 600이 됨), ja='bottom', ko='top'을 선택하여, 이것을 바탕으로 distinct words를 추출하고 있습니다. 만약 사용자가 더 적은 차별단어 목록을 원한다면, n=250으로 설정하여 더욱 정제된 단어 목록을 취득할 수 있습니다.

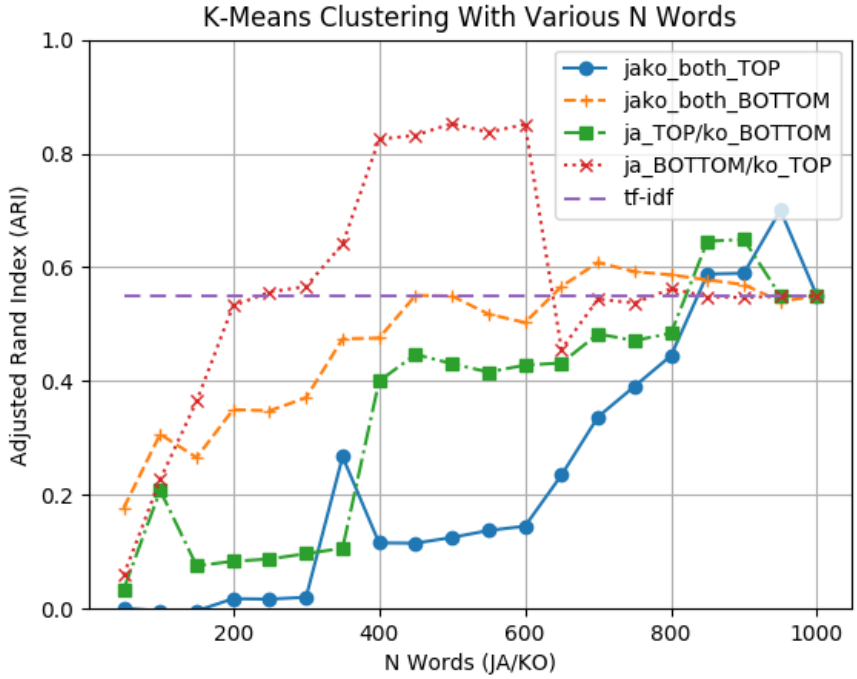

그림 2. 다양한 특징 단어 조합으로 클러스터링 성능을 비교한 결과

본 프로그램에 대해 궁금하신 점이 있으시거나, 이런 걸 같이 분석해 봤으면 좋겠다 등의 문의 사항이 있으시면 heeryon@gmail.com (조희련 앞)으로 메일 주시면 검토 후 답변 드리겠습니다.