KR-ESG
한국ESG연구소

네트워크 분석을 통한 중요인물 선정

-인물 데이터 분석 아이디어 공모전-

2023. 6. 16.

신의 환수 팀

목차

Ⅰ. 서론

인물 네트워크 분석을 통해 회사의 인물들 간의 관계를 이해하는 것은 현대 사회에서 중요한 리더십과 조직 운영에 대한 이해를 높이는 데 도움이 된다. 인물 네트워크를 구축하고 분석함으로써, 회사 내부에서 리더와 팔로워들 간의 상호작용과 권력 형태를 파악할 수 있다.

이를 위해 본 프로젝트는 회사 임원진들의 인물 네트워크를 구축하여 분석하였다. 여러 회사보다는 한 회사를 중점적으로 살펴보아 연도 별로 인물 네트워크가 어떻게 달라지는지 분석하고 추가적인 인사이트를 얻고자 하였다. 회사 설정은 분석 용이성을 위해 시장 상황에 민감한 증권사로 채택하였고, 증권사 중에서 대신증권을 채택하였다.

인물 네트워크는 각 인물을 노드로, 인물 간의 관계를 에지로 연결한다. 관계를 네트워크로 표현하기 위해서는 관계에 대한 정의가 먼저 이루어져야 한다. 사회 심리학적으로 유사한 특성의 사람들과 관계를 맺는 유사성의 법칙, 서로 다른 사람들과 관계를 맺는 상보성의 법칙, 상호 간의 이익을 극대화하기 위해 관계를 맺는 사회적 교환 이론 등의 이론이 존재한다.(김종운, 2017) 여기서 네트워크 형성에 보다 용이하게 적용할 수 있고, 다른 이론들보다 보편적으로 더 많은 영향을 끼친다고 알려진 유사성의 법칙에 기반해 인물 간의 관계를 인물 간의 유사성으로 치환해서 네트워크를 형성하고자 하였다.(Byrne, 1997) 즉, 노드의 연결 기준으로 두 노드 간의 유사성을 사용하였다.

본 분석의 목적은 중요 인물들을 추출하고 임원들의 네트워크를 구축하여 미처 파악하지 못한 중요 인물들을 발굴하거나 기존에 중요하다고 여겨졌던 인물들의 중요도를 재평가하는 것이다. 이를 통해 시장 상황과 네트워크의 변화 속에서 실제로 중요한 역할을 수행하는 인물들을 도출해낼 수 있으며, 조직의 효율성과 성과 향상, 전략 등에 기여할 수 있다.

Ⅱ. 분석 방법

1 진행순서

네트워크 구축과 분석은 다음과 같은 순서로 진행하였다.

1. 중요 인물 선정
   => 상황 속에서 변화한 인물을 중요 인물로 선정
   a. 상황의 변화
      i. 매출액의 증가
   b. 인물의 변화
      i. 주식 증가
      ii. 승진

2. 중요 인물과 다른 전체 인물들 사이의 네트워크 형성
   => 인물들을 노드로 삼고, 에지는 노드 간의 유사성을 가중치로 설정
   => 가중치(유사성, 연결강도)가 임계 값을 넘으면 두 노드 연결

3. 네트워크 분석

2 중요 인물 선정

중요 인물은 강한 영향력을 가지고 있어 네트워크의 구조나 연결성에 큰 영향을 미칠 수 있는 인물이다. 중요인물을 선정한다면 네트워크 분석 시 네트워크의 구조와 효율성, 인물 간의 영향력, 관계성의 흐름 등을 파악하기 더욱 쉬워진다. 이러한 이유로 네트워크 형성에 앞서, 회사 네트워크 내에서 영향력과 중요도가 강할 것이라고 예상되는 인물을 다음과 같은 기준으로 선정하였다.

중요인물은 상황의 변화에서 개인의 변화가 일어난 인물로 설정하였다. 상황의 변화는 매출이 증가하였을 때, 개인의 변화는 승진 또는 입사, 보유 주식량(의결권 있는 주식 수)의 증가로 기준을 설정하였다. 즉, 매출이 증가하였을 때 승진, 입사, 보유 주식량의 증가가 발생한 인물을 중요인물로 선정하였다. 예를 들어, 회사의 매출이 증가한, 즉 회사가 좋은 실적을 내고 있는 상황에서 승진을 한 인물은 회사의 실적에 기여를 한 영향력 있는 인물이라고 추론해볼 수 있을 것이다.

대신증권 재무 정보(연도별)

Aa 연도	# 연도별 누적 순이익(단위: 천...	Σ 이전 연도 대비 증...	Σ 증가율
2014	₩4,549,569	₩4,549,569	
2015	₩96,421,291	₩91,871,722	2119%
2016	₩30,578,147	-₩65,843,144	32%
2017	₩61,411,920	₩30,833,773	201%
2018	₩114,808,879	₩53,396,959	187%
2019	₩87,928,648	-₩26,880,231	77%
2020	₩170,364,113	₩82,435,465	194%
2021	₩178,657,494	₩8,293,381	105%
2022	₩86,474,236	-₩92,183,258	48%
2023	₩59,459,519	-₩27,014,717	69%
+ New			
NOT EMPTY 10	AVERAGE ₩89,065,382	AVERAGE ₩5,945,952	

표 1 대신증권 연도별 재무정보 대신증권의 연도별 매출액 변화를 살펴보기 위해 정리한 표이다.
* 부록 - 1. 데이터 부분 참조

조건에 해당하는 사람에 대해 연도별로 각각 조건이 맞을 때마다 카운트해 카운트 양이 2 이상일 경우 중요 인물로 판단하였다.
* 자세한 코드는 부록 - 2. 코드의 중요인물선정 부분 참조

위의 기준으로 선정한 중요인물은 다음과 같다.
선정한 중요인물: 양홍석, 이어룡, 홍대한, 송혁, 이순남, 김상원, 오익근, 김범철, 신인식, 조경순, 박성준, 정연규, 진승욱, 김성원, 권택현, 김호중, 이정화, 이재우, 박현식, 나유석, 신재범, 홍종국, 나재철, 박동현, 이동훈, 김병철, 정재중, 강윤기, 임민수, 정기동, 김수창, 최근영, 정연우, 강준규, 박규상, 하창룡, 장우철, 신재국, 김재중, 최명재, 배영훈, 이창세, 이지원, 문병식, 이성영, 김봉진, 길기모

단, 현재 절에서의 기준은 분석의 용이를 위해 설정한 임의의 기준으로 이러한 판단기준과 실제 중요 인물의 차이는 Ⅲ 절에서 더 살펴보고자 한다.

3 가중치 함수

가중치(유사성) 판단 기준은 대학, 전공, 직위, 근속연수, 담당업무, 경력, 보유 주식 수 총 7 가지 피처를 활용하였다. 선정한 피처의 기준은 인물이 보통 비슷한 환경에서 자라거나 지내오면 유사한 성질을 가지게 될 가능성 존재한다고 가정하고 환경적 요소에 중요한 영향을 미칠 것으로 판단되는 요소로 선정하였다. 예를 들어, 비슷한 전공의 사람들은 비슷한 분야에 대해 관심이 많고 비슷한 사고를 할 가능성이 높고, 비슷한 경력을 가진 사람들은 상황에 있어서 비슷한 대처 능력을 보일 가능성이 높다.

가중치 계산은 먼저, 각각의 피처에 대해 미리 정해둔 기준으로 노드 간의 거리를 구하고 α 값으로 1 보다 작은 양수를 설정해 두 노드 간의 각 피처의 유사성은 $\alpha^{거리}$ 로 구해주었다.(박경미, 2013) 두 노드 간의 전체 유사성은 각 피처의 유사성을 합해준 값으로 설정하였다. $weight(X,Y) = \sum_{i=1}^{7} \alpha^{거리_i}$ 본 프로젝트에서는 α값은 0.7 로 설정해주었다.
각 피처의 거리에 대한 기준은 아래와 같이 설정하였다.

3.1. 대학 거리 기준

먼저 대학은 기준을 지역으로 나누어서 아래와 같이 거리를 설정하였다.

a. 같은 대학: 0
b. 같은 지역의 대학: 1
c. 다른 지역의 대학: 2

이때, 대학 지역의 기준은 서울경기, 광역시, 그 외 한국, 해외의 4 가지 범주로 나누었다.

3.2. 전공 거리 기준

전공도 마찬가지로 범주로 나누어서 아래와 같이 거리를 설정하였다.

a. 같은 전공: 0
b. 같은 계열의 전공: 1
c. 다른 계열의 전공: 2

이때, 전공의 범주 기준은 대한민국의 학과 계열 분류를 참고해 경영경제, 사회과학, 법률, 인문, 기타, 자연, 공학의 7 가지 범주로 나누었다.

3.3. 직위 거리 기준:

직위의 거리는 각 계급을 순서대로 나열해서 범주를 나누고 거리를 설정하였다.

a. 같은 직위: 0
b. 사외이사를 제외하고 나머지는 각 노드 간의 계급 차이만큼 거리를 설정
c: 사외이사: 사외이사는 회사 밖 인물이므로 모두와 거리를 4 로 조정하였다.
(최대 거리와 최소 거리의 평균)

계급의 순서는 다음과 같다.

회장->대표이사->부회장->사장->부사장->전무->상무->상무보->이사대우->사외이사
대신증권의 계급도를 구하고 싶었으나 구하지 못해서 계급 순서는 일반적으로 적용되는 기준으로 임의로 설정하였다.

3.4. 근속연수 거리 기준

먼저 근속연수를 구하고 근속 연수의 차이를 5 년씩으로 잘라서 범주를 생성

a. 같은 근속연수: 0
b. 근속연수에 차이가 있으면 그 차의 범주 만큼을 거리로 설정

구체적으로, 근속연수 차이를 5 년 기준으로 범주를 나누어 각각 거리는 1 씩 증가한다.
(차이 1~5/6~10/11~15/...같은 그룹)

3.5. 담당업무 거리 기준

담당업무도 마찬가지로 범주를 나누어서 거리를 구하였다. 다만, 여기서는 같은 계열로 나누고 그 안에서 또 비슷한 업무로 범주를 2 번 나누었다. 예를 들어, IT 계열의 업무에는 IT 업무, 정보보호업무가 존재하고 IT 업무에는 IT 부문장, IT 본부장 등이 해당되게 된다.

a. 같은 업무: 0
b. 상위, 하위 혹은 비슷한 업무: 1
c. 같은 계열 업무: 2
d. 다른 계열 업무: 3

이때, 각 업무를 나누는 범주의 기준은 아래 대신증권의 직무소개를 참고하였다. (대신증권, n.d.)
개략적인 범주만 존재하고 자세한 업무의 분류는 존재하지 않아 세부적인 범주 설정은 위 사이트의 내용을 토대로 임의로 설정하였다.

3.6. 경력 거리 기준

경력을 구하기에 앞서 각 인물이 종사했었던 업무들의 범주를 나누었고 해당 인물의 경력 여부를 리스트로 만들어 각 업무에 종사했으면 1, 종사하지 않았으면 0 으로 설정하였다. 즉 리스트가 각 인물에 대한 경력 범주가 된다.
이때, 업무의 범주는 대신증권, 대신증권 자회사, 타증권사, 법·정책, 학교·교수·재단, 기타의 6 개의 범주로 나누었다.

a. 전체적으로 같은 업무 종사 (리스트 같음): 0
b. 법·정책, 학교·교수·재단, 기타에만 종사했었으면: 0.6*(1+같은 업무 종사한 개수)
+) 법·정책, 학교·교수·재단, 기타에만 종사하지 않았으면 이 세 범주는 기타로 통합
c. 두 인물이 종사했었던 업종 중 다른 업종이 존재: 다른 업종끼리의 최대, 최소 거리를 구한 후 평균을 거리로 삼음. 만약 같은 업종이 있을 시 거리 - (0.5*같은 업종 개수)

+) 각 업종 간 거리는 대신자회사->대신->타증권->기타 순으로 순서의 차이를 거리로 지정. 단, 기타의 경우 모든 업종과 거리 4 로 고정

인물 리스트 예시
인물 1: 대신증권, 대신증권 자회사, 타증권사 종사 ->[1, 1, 1, 0, 0, 0]
인물 2: 타 증권사, 기타 종사 -> [0, 0, 1, 0, 0, 1]

3.7. 보유 주식수 거리 기준

먼저 보유 주식수를 구하고 주식수의 차이를 100 씩으로 잘라서 범주를 생성

a. 같은 주식수: 0

b. 주식수에 차이가 있으면 그 차의 범주 만큼을 거리로 설정

이때, 각 인물 간에 주식수 차의 편차가 심해 주식수 하나만으로 유사성 전체에 큰 영향을 미칠 가능성이 존재해 로그 10 으로 로그 스케일을 추가적으로 진행해주어 거리를 설정하였다.

추가적으로 전체 가중치를 생성 후 가중치 간의 차이를 더욱 크게 보기 위해 각각 구한 가중치에 제곱을 해 스케일을 조정해주었다.

* 자세한 내용은 코드의 가중치함수 부분 참조

4 네트워크 형성

네트워크 구축에 앞서 각 연도 별로 임원진 데이터를 살펴보았을 때, 각각 임원진이 대개 30 명 전후로 존재하였다. 모든 임원진에 대해서 연결성을 살펴보면 약 30 개 전후의 노드들이 각각과 관계를 맺으면 30x30 의 가중치 행렬이 생성되게 된다. 이렇게 네트워크를 형성하면, 네트워크의 복잡성이 올라가 시각화를 해도 인사이트를 얻기 매우 어려워진다. 따라서 네트워크 형성 시 임계값을 기준으로 임계값 아래의 가중치를 가진 에지는 지웠고, 임계값 이상의 에지만 남겨 연결해주었다. 여기서 임계값은 가중치 분위수의 상위 10%로 설정하였다. 즉, 상위 10%의 가중치의 에지를 가지는 노드들만 연결되게 네트워크를 형성하였다. 추가적으로 앞서 선정한 중요인물과 비중요인물을 구별하기 위해 중요인물의 노드는 빨간색, 비중요인물의 노드는 파란색으로 설정해주었다. 네트워크 구축은 파이썬의 network 패키지를 사용하였다.

* 자세한 내용은 부록 - 2. 네트워크시각화 부분 참조

5 분석 지표

분석에 사용한 지표는 총 5 가지이다. (위키백과, 2022)

5.1 중요인물과 연결성

각 인물이 중요 인물과 얼마나 연결되어있는지 나타내주는 지표이다. 중요인물과 연결된 수를 나타내준다.

5.2 연결정도 중심성(Degree Centrality)

한 노드가 얼마나 많은 노드들과 관계를 맺고 있는지 나타내는 지표이다. 연결정도가 높은 노드는 다른 노드들과의 관계를 통해 정보 획득에 용이하며, 네트워크 내에 핵심노드일 가능성이 높다.

5.3 매개 중심성(Betweenness Centrality)

노드끼리 연결을 가장 빨리 해주는 경로 위에 있는 정도를 나타내며, 한 노드가 다른 노드들 사이에 위치하는 정도를 나타내는 지표이다. 매개 중심성이 높은 노드는 다른 핵심 노드들 사이에서의 중재자(broker) 역할을 수행하게 되며 핵심노드로서 기능할 가능성이 높다.

5.4 근접 중심성(Closeness Centrality)

각 노드 간의 거리를 바탕으로 중심성을 나타내는 지표이다. 다른 노드에 가장 빨리 연결하는 정도를 나타낸다. 연결정도 중심성은 직접 연결된 노드의 개수만을 파악한다면, 근접 중심성은 간접적으로 연결되어 있는 모든 노드간의 거리를 바탕으로 중심성을 측정한다. 간접적인 중심성이 크다면, 핵심 노드와의 관계가 많다는 뜻으로 다수의 핵심 노드들과의 관계를 통해 해당 네트워크 내의 핵심 노드일 가능성이 높다.

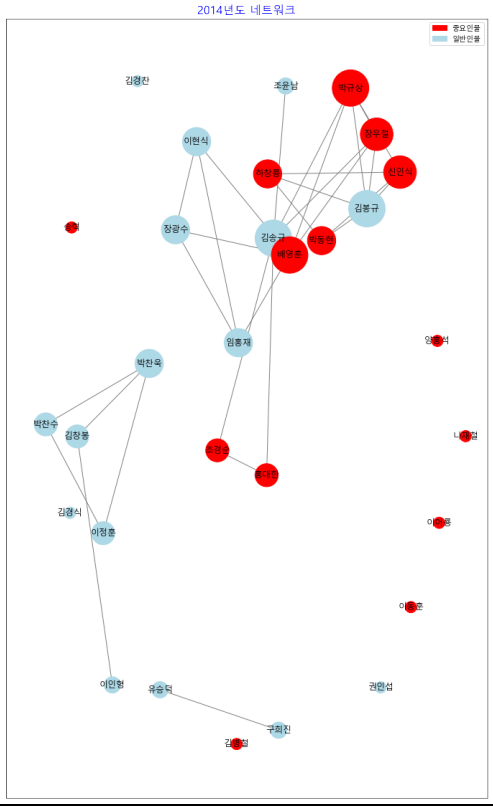
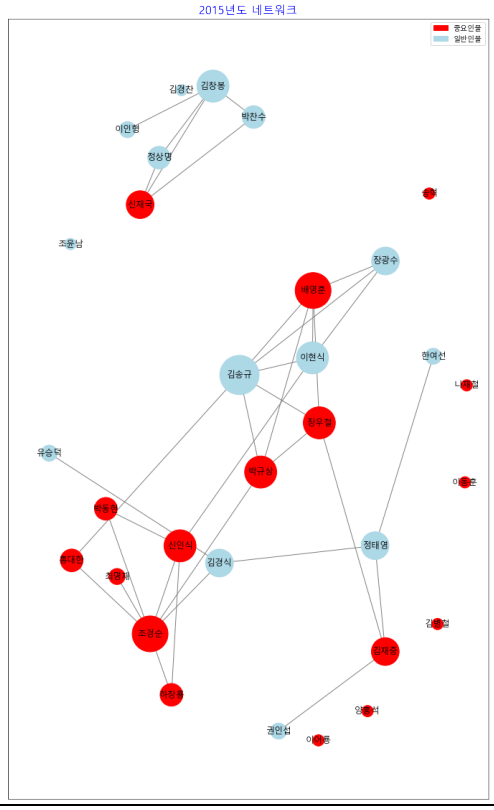
5.5 아이겐 벡터 중심성(Eigenvector Centrality)

한 노드와 연결된 다른 액터의 중심성을 가중치로 하여 계산하는 방식으로, 자신과 연결된 다른 노드들이 네트워크 내에서 얼마나 중요한지 파악하는 지표이다. 아이겐벡터 중심성이 높다면 연결정도가 적다 하더라도 높은 영향력을 가지며, 반대로 연결정도는 높지만 아이겐벡터 중심성이 낮다면 네트워크 내의 실질적인 영향력은 미미할 것으로 예측할 수 있다.

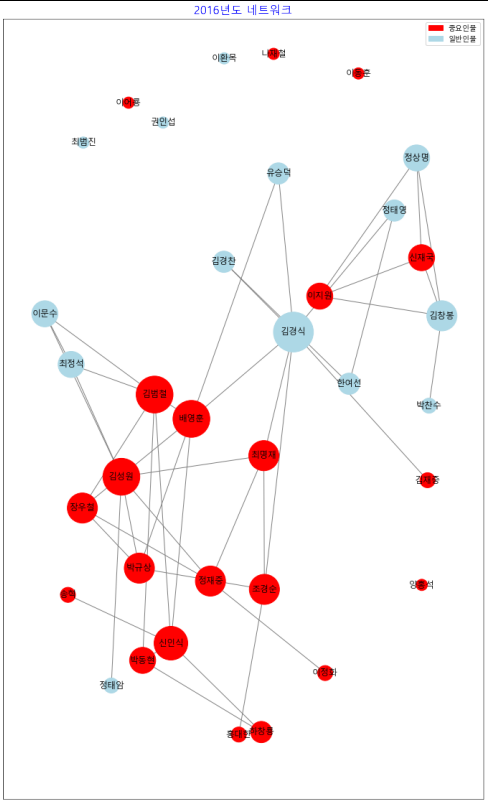
각 지표에 대해서 중요인물과 연결성은 직접 함수를 만들어 중요인물과 연결된 수를 count 해서 결과값으로 count 수를 나타내주었고, 나머지 중심성에 대한 지표들은 networkx 패키지의 중심성 구하는 함수들을 사용하였다.

Ⅲ. 분석

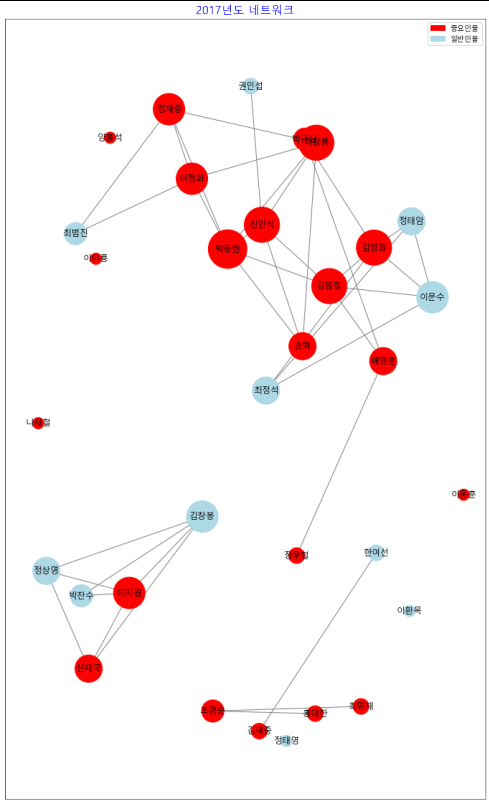
1 인물 네트워크

2014	2015
	

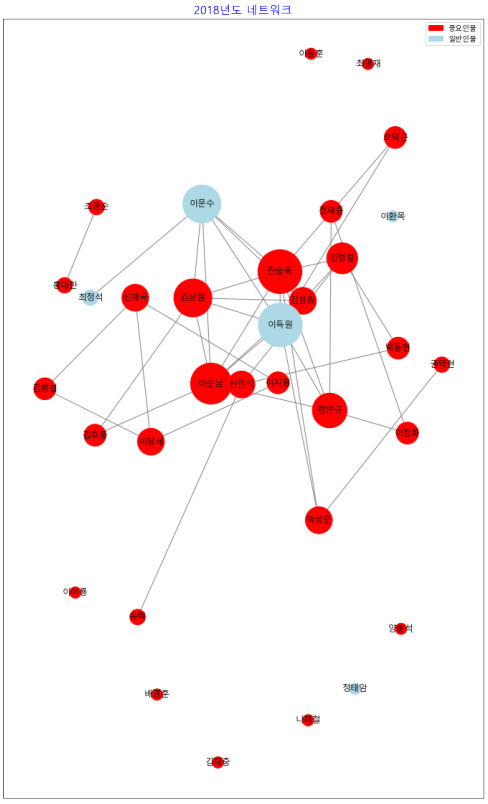
2016



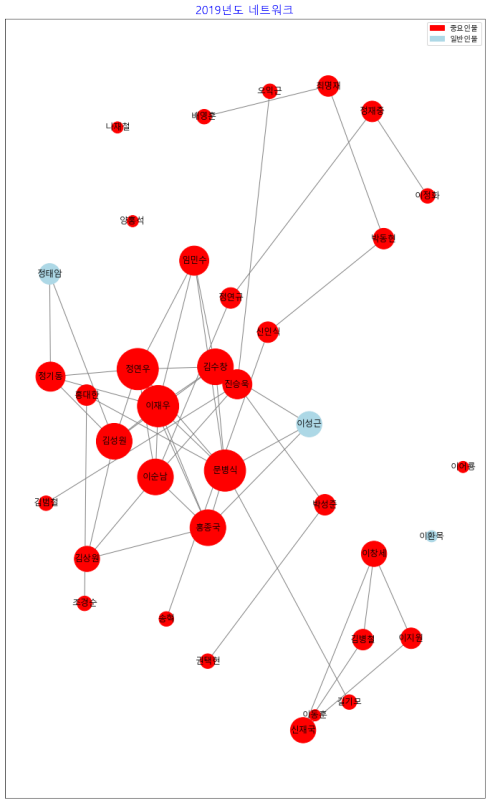
2017



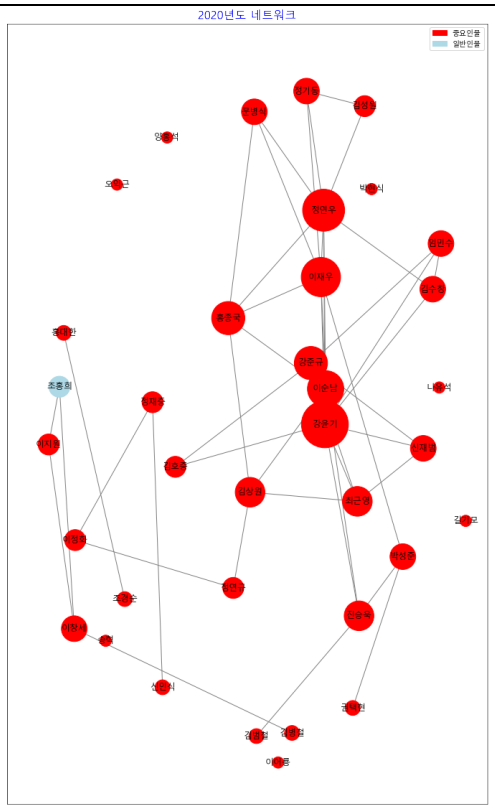
2018



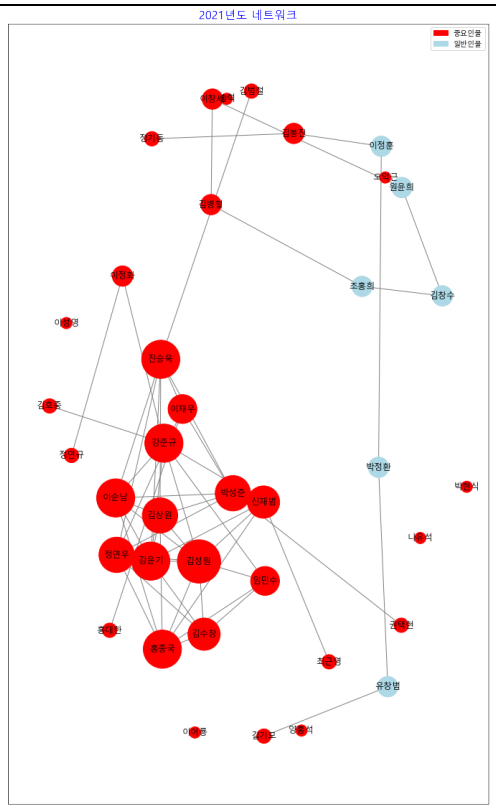
2019



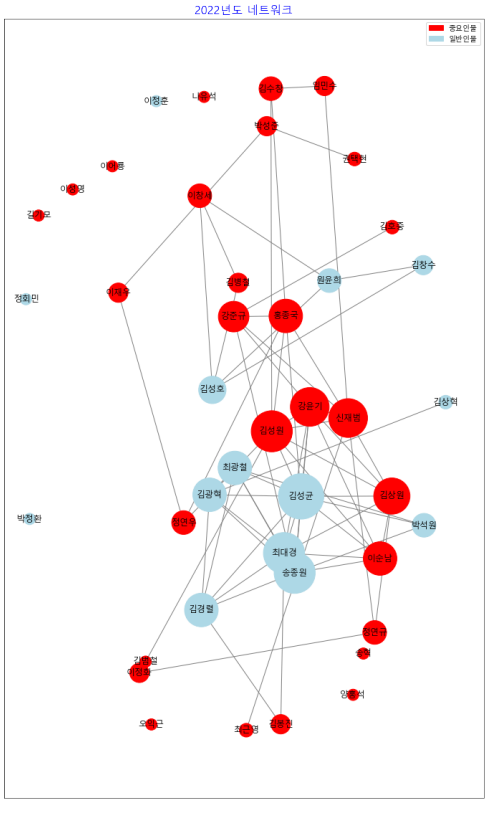
2020



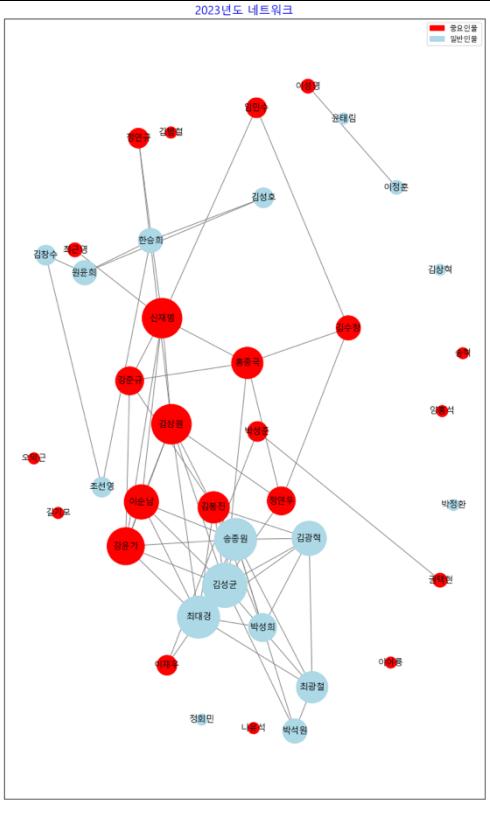
2021



2022



2023



2 연도별 인물의 중요도

2014

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
김봉규 5	박규상 0.172	김송규 0.078	박규상 0.277	박규상 0.427
김송규 4	김송규 0.172	박규상 0.076	장우철 0.264	김봉규 0.412
신인식 3	김봉규 0.172	배영훈 0.073	김송규 0.224	장우철 0.359
박규상 3	배영훈 0.172	장우철 0.042	배영훈 0.224	신인식 0.340
하창룡 2	신인식 0.137	김봉규 0.034	김봉규 0.215	배영훈 0.323
박동현 2	장우철 0.137	신인식 0.015	신인식 0.208	김송규 0.259
배영훈 2				하창룡 0.254
장우철 2				박동현 0.254

2015

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
김송규 4	김송규 0.200	조경순 0.126	조경순 0.283	김송규 0.480
조경순 4	배영훈 0.166	신인식 0.081	박규상 0.260	배영훈 0.448
박규상 3	조경순 0.166	김경식 0.076	김송규 0.253	박규상 0.357
장우철 3	박규상 0.133	김재중 0.062	배영훈 0.247	이현식 0.341
신인식 3	김창봉 0.133	장우철 0.061	김경식 0.240	장우철 0.339
하창룡 2	장우철 0.133	정태영 0.056	장우철 0.240	장광수 0.311
배영훈 2	신인식 0.133	김송규 0.038	신인식 0.240	조경순 0.192
박동현 2	이현식 0.133		홍대한 0.234	홍대한 0.164
이현식 2			이현식 0.229	신인식 0.155

2016

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
신인식 5	김경식 0.205	김경식 0.161	배영훈 0.338	배영훈 0.392
장우철 4	김범철 0.176	배영훈 0.126	김경식 0.323	김범철 0.363
박규상 4	김성원 0.176	신인식 0.074	최명재 0.290	장우철 0.295
김경식 4	배영훈 0.176	김성원 0.072	장우철 0.290	김경식 0.289
김범철 4	신인식 0.147	김범철 0.070	박규상 0.290	김성원 0.282
정재중 4		최명재 0.062	김범철 0.290	박규상 0.279
배영훈 4		조경순 0.042	김성원 0.279	신인식 0.260

2017

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
박동현 6	박동현 0.193	김범철 0.132	김범철 0.279	박동현 0.467
하창룡 5	김성원 0.161	박동현 0.078	박동현 0.259	하창룡 0.412
김범철 4	김범철 0.161	신인식 0.047	신인식 0.241	신인식 0.363
신인식 4	하창룡 0.161	김성원 0.041	김성원 0.219	이정화 0.316
송혁 3	신인식 0.161	배영훈 0.039	이문수 0.213	정재중 0.316
이정화 3		이문수 0.022	배영훈 0.201	송혁 0.292
정재중 3			하창룡 0.201	김범철 0.289

2018

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
이득원 7	진승욱 0.250	김범철 0.084	진승욱 0.334	진승욱 0.424
진승욱 6	이득원 0.250	진승욱 0.083	이득원 0.334	이득원 0.424
이순남 5	이순남 0.218	이득원 0.083	이순남 0.282	이순남 0.401
정연규 4	김상원 0.187	정연규 0.060	정연규 0.258	김상원 0.362
김상원 4	이문수 0.187	이문수 0.038	김상원 0.258	이문수 0.350
이문수 4	정연규 0.156	이순남 0.036	이문수 0.258	정연규 0.253
	김범철 0.125	박성준 0.032	김범철 0.250	김성원 0.206
		신인식 0.032		

2019

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
이재우 8	이재우 0.222	이순남 0.203	이재우 0.298	이재우 0.418
정연우 8	정연우 0.222	진승욱 0.122	정연우 0.298	정연우 0.418
문병식 7	문병식 0.222	문병식 0.097	이순남 0.298	문병식 0.354
이순남 6	홍종국 0.166	이재우 0.065	홍종국 0.280	김수창 0.332
홍종국 5	김수창 0.166	정연우 0.065	문병식 0.253	홍종국 0.304
김수창 5	김성원 0.166	정연규 0.063	김상원 0.248	김성원 0.280
김성원 5	이순남 0.166	홍종국 0.044	김수창 0.244	임민수 0.265

2020

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
강윤기 10	강윤기 0.285	강윤기 0.125	이순남 0.307	강윤기 0.435
정연우 8	정연우 0.228	김상원 0.124	강윤기 0.307	정연우 0.397
이재우 7	이재우 0.199	정연규 0.095	이재우 0.288	이재우 0.367
이순남 6	이순남 0.171	이순남 0.072	정연우 0.288	이순남 0.345
홍종국 5	홍종국 0.142	이정화 0.067	홍종국 0.271	홍종국 0.279
강준규 5	강준규 0.142	정연우 0.066	김상원 0.265	강준규 0.199
진승욱 4		이재우 0.064	진승욱 0.260	문병식 0.198
김상원 4			최근영 0.260	
최근영 4				

2021

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
김성원 9	김성원 0.243	강준규 0.079	김성원 0.295	김성원 0.402
강준규 7	강준규 0.189	진승욱 0.043	이순남 0.286	이순남 0.340
진승욱 7	진승욱 0.189	이순남 0.036	강윤기 0.286	강윤기 0.330
이순남 7	이순남 0.189	박성준 0.033	강준규 0.271	홍종국 0.320
강윤기 7	강윤기 0.189	김성원 0.315	진승욱 0.263	김상원 0.304
홍종국 7	홍종국 0.189	정연우 0.0304	정연우 0.263	진승욱 0.262
				강준규 0.256

2022

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
김성원 8	김성균 0.268	김성원 0.095	김성원 0.324	김성균 0.402
신재범 8	김성원 0.219	김성균 0.088	김성균 0.317	최대경 0.343
김성균 5	최대경 0.219	정연우 0.080	홍종국 0.311	송종원 0.335
홍종국 5	송종원 0.219	신재범 0.062	강윤기 0.311	강윤기 0.317
김상원 5	강윤기 0.195	이재우 0.056	김상원 0.298	이순남 0.276
강윤기 5	신재범 0.195	홍종국 0.054	강준규 0.282	김상원 0.266
	김상원 0.170	강준규 0.035	이순남 0.282	김성원 0.256

2023

중요인물과 연결	Degree	Betweenness	Closeness	Eigenvector
신재범 8	김성균 0.256	김상원 0.079	김상원 0.305	김성균 0.391
김상원 6	송종원 0.230	정연우 0.077	김성균 0.282	최대경 0.380
김성균 5	최대경 0.230	신재범 0.054	홍종국 0.275	송종원 0.351
강윤기 4	신재범 0.205	이재우 0.051	강윤기 0.269	강윤기 0.306

정연우 4	김상원 0.205	김성균 0.049	최대경 0.269	김상원 0.289
최대경 4	강윤기 0.179	홍종국 0.038	이순남 0.262	이순남 0.287
홍종국 4	이순남 0.153	박성준 0.026	신재범 0.262	김광혁 0.264
	김광혁 0.153	송종원 0.021	강준규 0.251	최광철 0.230
			정연우 0.251	

* 현재 절의 연도별 인물의 중요도는 상위인물들만 추출해온 것으로, 전체 인물에 대한 결과는 부록 - 2. 코드를 실행한 결과를 참조

3 연도별 네트워크 변화

Ⅱ. 분석방법의 2절에서 선정한 중요 인물의 기준이 연도별 회사 매출의 증가에 따른 개인의 변화였기 때문에, 연도가 지날수록 회사에 오래 남아있는 사람들은 회사의 매출의 증가를 더 많이 겪었기 때문에 자연히 중요인물이 될 수밖에 없다. 위 결과의 네트워크 그래프에서도 볼 수 있듯이 연도가 지날수록 중요인물(빨간색 노드)이 많아지는 것을 확인할 수 있었다. 특히 18, 19, 20, 21년도에 중요인물이 많이 증가한 것을 확인할 수 있었는데, 이 해에 매출액이 증가한 것이 영향을 미치는 것으로 추정할 수 있었다. 21년도 이후 다시 비중요인물의 수가 늘어난 것을 확인할 수 있었는데, 이는 21년도 이후에 매출액 증가가 없어서 네트워크에 새로 유입된 사람은 중요인물로 선정되지 않은 것으로 추정한다. 또한 보통 전년도에 중요도 지표 값이 높았던 사람들이 다음 해에도 높은 중요도 값을 유지하는 패턴을 발견할 수 있었다.

4 분석 결과

Ⅲ. 분석결과 2절의 연도별 인물의 중요도를 살펴보면, 중요한 인물과 연결된 여부의 지표값이 높은 사람일수록 전체적으로 다른 지표의 값들이 높게 나와 네트워크 상의 중요도가 높은 사람임을 확인할 수 있었다. 이를 통해 처음에 선정한 중요 인물 선정 기준이 유의미하다고 판단할 수 있으며, 매출액의 증가에 따른 개인의 변화가 회사 내에서의 인물의 중요도를 판단할 수 있는 간략한 지표로서 작동할 수 있다는 점을 확인할 수 있었다.

중요 인물들과 연결이 많이 된 인물들에 대해서도 추가로 분석을 진행해보았다. 연도별로 중요한 인물과 연결된 여부 지표값이 높은 상위 5명을 뽑아서 그 인물의 특성을 살펴보았는데, 모두 대신증권에서 일한 경험이 있던 사람들이었으며, 그 중 거의 모든 사람들은 대신증권에서만 일했던 사람들임을 확인할 수 있었다. 겹치는 인물들을 제외하고 총 24명의 인물들 중 4명빼고 83%의 사람들은 전부 대신증권에서만 일했던 사람들이며, 그 4명 역시 대신증권에서 일했던 경험이 있으며 김범철, 진승욱, 이득원과 같은 사람들은 대신증권과 대신증권의 자회사에서만 일했던 사람들이며 김경식의 경우만 대신증권과 타증권사에서 일했던 경험을 가지고 있는 사람임을 확인할 수 있었다. 이를 통해 대신증권에서만 일했던 요소가 인물네트워크에서의 영향력에 영향을 미칠 수 있다는 점을 확인할 수 있었다. 이와 같이 분석 결과로 도출된 중요 인물들의 공통점을 파악하면 인물의 영향력을 파악하는 데 활용할 수 있을 것으로 사려 된다.

이어룡, 양홍석과 같이 직급이 높은 분들은 다른 인물들과 연결된 노드가 없고, 연결이 많은 대부분의 인물들은 중간 직급에 속하는 사람들인 것을 확인할 수 있었다. 이는 최상위 계급은

다른 인물들과 유사성이 적으며, 이 유사성을 관계라 한다면 다른 인물들과 맺고 있는 관계가 적을 것이며, 중간 직급들은 활발한 관계를 갖고 있다고 볼 수 있으며 이러한 지배구조를 가지고 있을 것이라고 추측할 수 있을 것 같다.

분석결과의 연도별 네트워크 변화에서 전년도에 중요도 지표값이 높았던 사람들이 다음 해에도 높은 중요도 값을 유지하는 패턴을 발견할 수 있었는데, 이는 중요 인물들이 영향력과 권력을 유지하여 중요한 역할을 수행하는 것이라 판단할 수 있을 것 같다.

아래 표는 분석 결과에서 중요도 지표 값들이 높은 인물들을 연도별로 정리한 것이다.

연도	이름
2014	박규상
2015	김송규, 조경순
2016	신인식, 김경식, 배영훈
2017	박동현, 김범철, 하창룡
2018	이득원, 진승욱, 이순남
2019	이재우, 정연우, 문병식
2020	강윤기, 정연우, 이재우
2021	김성원, 강준규, 이순남
2022	김성원, 김성균
2023	김성균, 김상원, 송종원

표2 연도별 중요인물

Ⅳ. 결론 및 고찰

여태까지 유사성을 기준으로 인물네트워크를 형성해 네트워크를 분석해보았다. 설정한 지표에 따라 네트워크에서 영향력이 있는 중요인물을 선정할 수 있었으며, 분석 전 설정한 중요 인물에 대한 지표가 유의미할 수도 있다는 점을 확인할 수 있었다. 또한 네트워크의 형태로 회사의 지배구조에 대해서도 생각해볼 수 있었으며, 연도별 지표값에 따른 네트워크의 패턴에 대해서도 살펴볼 수 있었다. 또한, 지표에 선정된 상위 사람들의 특성이 어느정도 유사할 수 있음을 확인해볼 수 있었다.

중요인물 선정을 토대로 중요인물과 비중요인물들 간의 관계를 파악하고, 네트워크 구조에서의 역할과 영향력을 분석하였다. 이를 통해 비중요인물들 중에서 중요인물들과 많이 연결되어 두각을 드러내는 인물을 발굴해낼 수 있었으며, 또한 중요인물들 중에서 네트워크의 중심에 있는 인물들을 분석할 수 있었다.

중요 인물들의 식별과 네트워크 분석을 통해 조직 내에서 영향력과 권력을 가진 인물들과 어느 시기에 어떤 네트워크가 활성화되는지 파악할 수 있으며 이는 리더들이 어떤 인물들을 중심으로 네트워크를 형성해야 하는지에 대한 결정에 도움을 줄 수 있다. 또한 조직의 변화와 동향을 파악하고 리더십의 효과를 평가를 통해 향후 전략과 의사결정에 반영할 수 있다.

또한 추가적인 지표도 고려해 볼 수 있다. 본 분석에서는 유사성에만 국한하여 네트워크를

형성하였고, 중요도 판단 지표 또한 다양성이 부족하다고 판단할 수 있다. 이에 추가적으로 크롤링을 통해 기사에서 이름이 묶여 나오는 횟수로 인물 간의 관계에 대한 특성과 기사수를 인물의 사회적 영향력으로 잡아서 네트워크를 형성하고 분석해보는 방법도 고려해볼 수 있을 것이다. 즉, 더 많은 데이터를 활용하여 유사성 외 다른 연결 관계를 추가하고, 사회적 평판을 판단할 수 있는 데이터를 활용해 중요도 지표를 추가할 수 있을 것이다. 다양한 지표를 활용하면 보다 정확하고 포괄적인 네트워크 분석이 가능해질 것으로 예상한다.

VI. 참고자료

김종운. (2017). 만남 그리고 성장을 위한 인간관계 심리학. 학지사.

대신증권(n.d.). 직무소개. http://money2.daishin.co.kr/company/recruit/job_introduce.shtml

박경미, 김성환 and 조환규. (2013). 소설 등장인물의 텍스트 거리를 이용한 사회 구성망 분석. 한국콘텐츠학회 논문지, 13(4), 427-439.

위키백과(2022). 사회 연결망 분석. https://ko.wikipedia.org/wiki/사회_연결망_분석

Byrne, D. (1997). An overview (and underview) of research and theory within the attraction paradigm. Journal of Social and Personal Relationships, 14(3), 417-431.

VII. 부록

1. 데이터

데이터는 DART(전자 공시 시스템)에 기재된 데이터를 사용하였다. DART 에 openDartAPI 가 존재해 공시정보를 바로 파이썬으로 받아올 수 있지만, 이렇게 받아온 데이터에는 임원진의 '의결권 있는 주식수'가 누락 되어있고, 연도도 한정되어 있었다. 이러한 이유로 따로 노션(Notion)에 데이터베이스를 직접 만들어 파이썬으로 가져오는 방식을 사용하였다.

재무분석

연도별

대신증권 재무 정보(연도별)

Aa 연도	# 연도별 누적 순이익(단위: 천...	Σ 이전 연도 대비 증...	Σ 증가율
2014	₩4,549,569	₩4,549,569	
2015	₩96,421,291	₩91,871,722	2119%
2016	₩30,578,147	-₩65,843,144	32%
2017	₩61,411,920	₩30,833,773	201%
2018	₩114,808,879	₩53,396,959	187%
2019	₩87,928,648	-₩26,880,231	77%
2020	₩170,364,113	₩82,435,465	194%
2021	₩178,657,494	₩8,293,381	105%
2022	₩86,474,236	-₩92,183,258	48%
2023	₩59,459,519	-₩27,014,717	69%
+ New			
NOT EMPTY 10	AVERAGE ₩89,065,382	AVERAGE ₩5,945,952	

분기별

대신증권 재무 정보(분기별)

Aa 연도	분기	# 분기별 순이익(단위: 천원)	Σ 이전 분기 대비 증가량	Σ 증가율
2014	4분기	₩4,414,000	₩4,414,000	
2015	1분기	₩37,095,677	₩32,681,677	840%
2015	2분기	₩32,930,656	-₩4,165,021	89%
2015	3분기	₩23,744,046	-₩9,186,610	72%
2015	4분기	₩2,650,912	-₩21,093,134	11%
2016	1분기	₩25,251,755	₩22,600,843	953%
2016	2분기	₩669,450	-₩24,582,305	3%
2016	3분기	₩7,748,564	₩7,079,114	1157%
2016	4분기	-₩3,091,622	-₩10,840,186	-40%
2017	1분기	₩31,467,655	₩34,559,277	-1018%
2017	2분기	₩7,520,735	-₩23,946,920	24%
2017	3분기	₩27,142,372	₩19,621,637	361%
2017	4분기	-₩4,718,842	-₩31,861,214	-17%
2018	1분기	₩59,616,952	₩64,335,794	-1263%
2018	2분기	₩43,341,501	-₩16,275,451	73%
2018	3분기	₩21,363,917	-₩21,977,584	49%
NOT EMPTY 34		AVERAGE ₩26,191,713	AVERAGE ₩1,748,809	

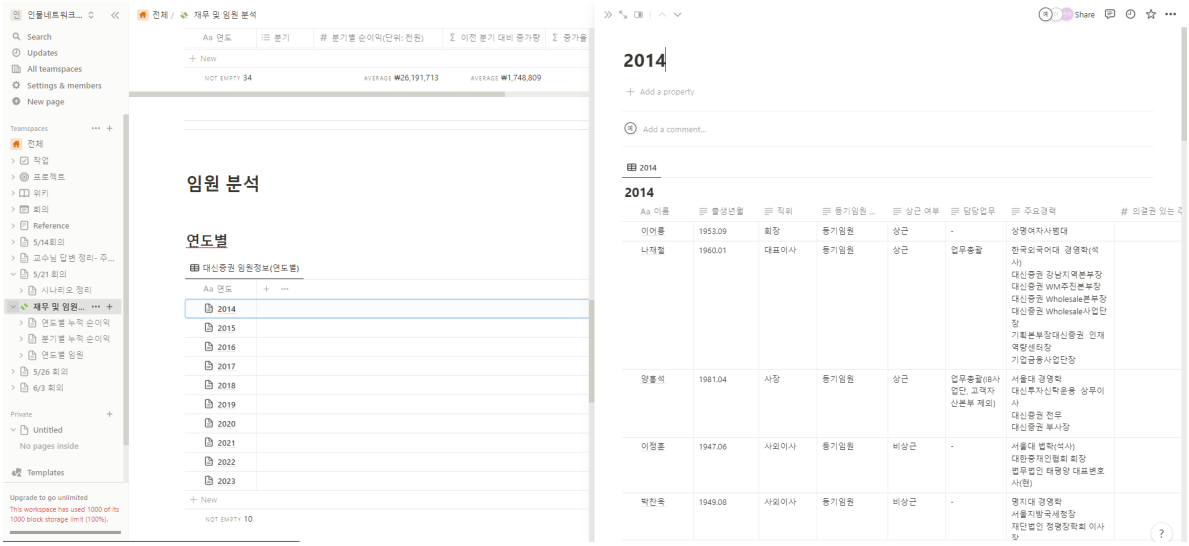


그림 1. 노션 DB 정리 예시

2. 코드

```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import re
import math

import requests, json
import pandas as pd

NOTION_TOKEN = "secret_Alc5tovrF5rnoYWC54z9CzwQUNb5d2A1q3HPKwOEqup"
DATABASE_ID = "213b4ff8bb884cfc987139ae53aa4396"

headers = {
    "Authorization": "Bearer " + NOTION_TOKEN,
    "Content-Type": "application/json",
    "Notion-Version": "2022-06-28",
}
```

```
DATABASE_ID_LIST = ["db2b026a03f0442f828bc88dcdbf143c",
"c238334399d54b10aad0bcb84cb77d5c", "bf7efa466c624f1ca200d05c5cf34f58",
"2f0e230f82974dd09841ae871cb558f9", "b708c8576ea34bb385d24aedc5b1277b",
"d6211c2113914ecc86332d4bb305a115", "a8b0289c72eb4daaa1063152eb5e787a",
"73e5dc61f97f4f2095c0265f2dae70d8", "1a91276509b246dda4c170a248d096c2",
"6a374753947e42f7aa57fac3a474b78f"]
YEAR_LIST = [2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023]

def readDatabase(DATABASE_ID_LIST, YEAR_LIST, headers):
    for id in zip(DATABASE_ID_LIST, YEAR_LIST):
        readUrl = f"https://api.notion.com/v1/databases/{id[0]}/query"
        res = requests.request("POST", readUrl, headers=headers)
        data = res.json()
        print(res.status_code)

        if res.status_code == 200:
            try:
                with open(f"./{id[1]}.json", 'w', encoding='utf8') as f:
                    json.dump(data, f, ensure_ascii=False)
            except:
                continue

readDatabase(DATABASE_ID_LIST, YEAR_LIST, headers)

def jsonToDataFrame(YEAR_LIST):
    temp = {}
    year_list = []
    name = []
    birth = []
    registered = []
    fullTime = []
    responsibilities = []
    career = []
    stock = []
    time1 = []
    time2 = []
    position = []

    for year in YEAR_LIST:
        with open(f"./{year}.json", 'r') as f:
```

```
                json_data = json.load(f)
                for i in range(len(json_data['results'])):
                    properties = json_data['results'][i]['properties']
                    year_list.append(year)
                    for key in properties:
                        if key == '이름':
                            name.append(properties[key]['title'][0]['text']['content'])
                        elif key == '출생년월':
                            birth.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '등기임원 여부':
                            registered.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '상근 여부':
                            fullTime.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '담당업무':

responsibilities.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '주요경력':
                            career.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '의결권 있는 주식수':
                            stock.append(properties[key]['number'])
                        elif key == '재직기간':
                            time1.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '임기만료일':
                            time2.append(properties[key]['rich_text'][0]['text']['content'])
                        elif key == '직위':
                            position.append(properties[key]['rich_text'][0]['text']['content'])

    temp['이름'] = name
    temp['연도'] = year_list
    temp['출생년월'] = birth
    temp['등기임원 여부'] = registered
    temp['상근 여부'] = fullTime
    temp['담당업무'] = responsibilities
    temp['주요경력'] = career
    temp['의결권 있는 주식수'] = stock
    temp['재직기간'] = time1
    temp['임기만료일'] = time2
    temp['직위'] = position
    return temp
```

```
dict = jsonToDataFrame(YEAR_LIST)

df = pd.DataFrame.from_dict(data=dict, orient='columns')



# In[2]:



df['의결권 있는 주식수'].fillna(0.0, inplace=True)
#'₩xa0' 처리
df['주요경력']=df['주요경력'].str.replace('₩xa0', '')
#대학, 전공 피처 생성
temp=df['주요경력'].str.split('₩n')
colleges=temp.str.get(0)

#추가로 처리해줘야될 대학 form 맞춰주기
colleges[colleges== 'University of Bath 사회과학대학원(석사)'] ='UniversityofBath
사회과학대학원(석사)'
colleges[colleges== 'Florida International University
경영대학원(박사)']='FloridaInternationalUniversity 경영대학원(박사)'
colleges[colleges== 'Ohio State University 정책대학원(박사)'] = 'OhioStateUniversity
정책대학원(박사)'
colleges[colleges== 'KDI 국제정책대학원(석사)'] = 'KDI 국제정책대학원(석사)'
colleges[colleges== '상명여자사범대'] = '상명여자사범대 교육학'
colleges[colleges== '한국외국어대경영학(석사)'] = '한국외국어대 경영학(석사)'
colleges[colleges== '미시간대학교 경영대학원 MBA'] = '미시간대학교 경영대학원 MBA'

#대학, 전공으로 나누기
ttemp = colleges.str.split()

#대학 피처 생성
df['대학교']=ttemp.str.get(0)

#전공 피처 생성
tttemp = ttemp.str.get(1)
tttemp=tttemp.replace(r'₩([^)]*₩)', '', regex=True)
df['전공']=tttemp

#근속연수 생성
def create_work_years(x):
```

```
    temp = re.split(r'~|~₩n', x)
    temp[0] = temp[0].strip()
    temp[1] = temp[1].strip()
    return int(temp[1][:4]) - int(temp[0][:4])

df['근속연수'] = df['재직기간'].apply(lambda x : create_work_years(x))

#주요경력 전처리
def replace_career(x):
    temp = [string.split() for string in re.split(r'₩n |₩n', x)]
    if len(temp)==1 :
        return {'대신증권'}
    else:
        return set(list(zip(*temp[1:]))[0])

df['주요경력요약'] = df['주요경력'].apply(lambda x : replace_career(x))


# In[3]:


df_2014 = df[df['연도']==2014].reset_index(drop=True)
df_2015 = df[df['연도']==2015].reset_index(drop=True)
df_2016 = df[df['연도']==2016].reset_index(drop=True)
df_2017 = df[df['연도']==2017].reset_index(drop=True)
df_2018 = df[df['연도']==2018].reset_index(drop=True)
df_2019 = df[df['연도']==2019].reset_index(drop=True)
df_2020 = df[df['연도']==2020].reset_index(drop=True)
df_2021 = df[df['연도']==2021].reset_index(drop=True)
df_2022 = df[df['연도']==2022].reset_index(drop=True)
df_2023 = df[df['연도']==2023].reset_index(drop=True)


# In[4]:


#학교 거리 계산
def cal_college(df1, df2, alpha = 0.7):
    #지역별로 나눔
```

```
    college_dict = {'서울경기':['상명여자사범대', '서울대', '홍익대', '성균관대', '연세대',
'연세대학교', '동국대', '중앙대', '한국외대', '한국외국어대', '세종대학교', '세종대',
'성균관대학교', '경희대', '서강대', '고려대', '건국대', '가천대', '가천대학교', '한양대', '명지대'],
                            '광역시':['부경대', '경북대', '전남대', '조선대', '한국과학기술원', '가천대',
'인하대', '동아대', '대전대', '울산대', '영남공업전문대'],
                            '그외한국':['전북대', '원광대', '충북대', '충북대학교', '강원대', 'KDI'],
                            '해외':['FloridaInternationalUniversity', 'OhioStateUniversity',
'UniversityofBath', '조지아공대', 'Brown 대', '카네기멜론대', '미시간대학교']}

    #키값 찾기
    for key, val in college_dict.items() :
        if df1['대학교'] in val :
            df1_key = key
        if df2['대학교'] in val :
            df2_key = key

    #k 값 구하기
    if df1['대학교'] == df2['대학교'] :
        k=0
    elif df1_key == df2_key :
        k=1
    elif df1_key != df2_key :
        k=2

    return alpha ** k

#학과 거리 계산
def cal_department(df1, df2, alpha = 0.7):
    department_dict={'경영경제':['회계학', '세무학', '경제학', '국제경영학', '경영학', '무역학',
'MBA', '경제통상학', '경영대학원', '경영대학원 MBA', '산업정보학', '금융공학', '국제금융학'],
                            '사회과학':['사회학', '사회과학대학원', '행정학', '정치외교학', '정치학',
'신문방송학'],
                            '법률':['법학', '사법학', '정책대학원', '국제정책대학원'],
                            '인문':['불어불문학'],
                            '기타':['교육학', '인적자원개발학'],
                            '자연':['응용통계학', '통계학', '수학'],
                            '공학':['기계공학', '화학공학', '전자계산학', '전자전산학', '전산학',
'항공우주학']}

    #키값 찾기
```

```
    for key, val in department_dict.items() :
        if df1['전공'] in val :
            df1_key = key
        if df2['전공'] in val :
            df2_key = key

    #k 값 구하기
    if df1['전공'] == df2['전공'] :
        k=0
    elif df1_key == df2_key :
        k=1
    elif df1_key != df2_key :
        k=2

    return alpha ** k

#직위 거리 계산
def cal_position(df1, df2, alpha = 0.7):
    position_dict={'회장':0, '대표이사':1, '부회장':2, '사장':3, '부사장':4, '전무':5, '상무': 6,
'상무보':7, '이사대우부문장':8, '이사대우담당':8, '사외이사':99, '감사위원':99}

    #value 값 찾기
    df1_value = position_dict[df1['직위']]
    df2_value = position_dict[df2['직위']]

    #k 값 구하기
    if df1['직위'] == df2['직위'] :
        k=0
    elif df1_value == 99 or df2_value == 99 : #사외이사 혹은 감사인원(회사밖 인물)은
모두와 거리 4(최대차이의 중간값// 회장, 이사대우담당과의 거리)
        k=4
    else :
        k=abs(df1_value - df2_value) #값 차이만큼 거리

    return alpha ** k

# 근속연수 거리 계산
def cal_work_years(df1, df2, alpha = 0.7):
    #k 값 구하기
    if df1['근속연수'] == df2['근속연수'] :
```

```
        k=0
    else :
        if abs(df1['근속연수'] - df2['근속연수']) % 5 ==0:      #근속연수 차이 5 년 기준으로 1 씩 증가 (차이 1~5/6~10/11~15/...같은그룹)
            k= abs(df1['근속연수'] - df2['근속연수']) // 5
        else:
            k= 1 + abs(df1['근속연수'] - df2['근속연수']) // 5

    return alpha ** k

#담당업무 거리 계산
def cal_task(df1, df2, alpha = 0.7):
    task_dict={'IT 계열': {'IT':['IT 부문장', 'IT 본부장', 'IT 서비스본부장', '디지털부문장', '스마트 Biz 본부장', 'Operation&Technology 본부장'], '정보보호':['정보보호부문장', '정보보호담당']},
                    '본사영업':{'Wolesale':['Wholesale 부문장','Wholesale 영업본부장', 'Wholesale 사업단장', 'Trading 부문장', 'Trading Center 장', 'Sales&Trading 총괄', '대외협력담당'],
                                'IB':['IB 사업단장', 'IB 부문장', 'IB 부부문장', 'IB1 부문장', 'IB2 부문장', 'IPO 담당',  'PF 부문장', 'PF1 본부장', '구조화상품본부장'],
                                '영업':['영업부장']},
                    '본사관리':{'리테일':['리테일총괄', 'Club1962 센터장','WM 추진부문장', 'WM 추진본부장', 'WM 사업단장'],
                                '리스크':['리스크관리부문장', '리스크관리담당'] ,
                                '경영':['경영전략총괄', '전략지원부문장', '전략지원담당', '경영지원본부장', '경영지원부문장', '경영기획부문장', '경영기획본부장', '기획본부장', '기획본부장 / 투자금융담당', '전략지원부문장/프라이빗라운지 부문장'],
                                '상품':['Product 부문장', 'Solution&Product 사업단장'],
                                '업무':['업무총괄',  '업무총괄(IB 사업단, 고객자산본부 제외)', '업무총괄(IB 사업단,고객자산본부 제외)', '업무총괄 (IB 사업단,고객자산본부 제외)', '업무총괄(IB 사업단, 고객자산본부, 경영지원본부, 정보보호부문, 준법지원부문, 감사부문 제외)'],
                                '금융':['기업금융담당', '스마트금융본부장', '금융주치의추진본부장', '금융주치의사업단장'],
                                '인사,인프라':['인재역량센터장', '인프라관리본부장'],
                                '법률':['준법감시인₩n 준법지원부문장', '준법감시인 준법지원부문장', '준법감시인₩n 준법지원부문장', '준법감시인₩n 준법지원담당', '준법감시인/준법지원부문장', '준법감시인 / 준법지원담당', 'Advisory 본부장'],
                                '감사':['감사위원장', '감사부문장', '감사위원', '상근감사위원', '감사담당'],
```

```
                    '언론홍보':['홍보부문장', 'Coverage 본부장']},
                '리서치': {'리서치센터':['Research Center 장', '리서치센터장', 'Market₩n
Solution 부문장', 'Research&Strategy 본부장', 'Market Solution 부문장', 'Market Solution
부문장']},
                '고객':{'고객자산':['고객자산부문장', '고객자산부문  부부문장', '고객자산본부장',
'고객자산본부장 / 홍보담당', '고객자산본부장/ 홍보담당', '고객자산본부장/ 홍보부문장',
'고객자산본부장/홍보부문장', '고객자산부문장, 홍보부문장', '고객자산부문장₩n 홍보부문장'],
                    'WM':['서부 WM 부문장', '서부 WM 본부장', '재경 1WM 부문장',
'재경 1WM 본부장', '재경 2WM 본부장', '재경 2WM 부문장', '동부 WM 부문장',
'동부 WM 본부장'],
                    '프라이빗':['프라이빗부문장,₩n  나인원프라이빗라운지장',
'프라이빗부문장, 나인원프라이빗라운지장', '대신나인원 ₩n 프라이빗라운지장',
'프라이빗라운지  부문장 /대신나인원  프라이빗라운지장', '프라이빗부문장,
대신나인원프라이빗라운지장', '프라이빗부문장₩n 나인원프라이빗라운지장',],
                    '소비자보호':['금융소비자보호부문장₩n(COO)',
'금융소비자보호부문장(COO)', '금융소비자보호부문장  (COO)', '금융소비자보호  총괄',
'금융소비자보호총괄']},
                '기타':{'지점':['강북지역본부장', '강남지역본부장', '강남선릉센터장',
'서부지역본부장', '동부지역본부장', '울산지점장'],
                    '비서':['비서실장', '비서/브랜드본부', '비서/브랜드담당'], '공란':['-']}}


    #키값 찾기
  for outerKey in task_dict.keys() :
    for key, val in task_dict[outerKey].items() :
      if df1['담당업무'] in val :
        df1_outerKey = outerKey
        df1_innerKey = key
      if df2['담당업무'] in val :
        df2_outerKey = outerKey
        df2_innerKey = key


  #k 값 구하기
  if df1['담당업무'] == df2['담당업무'] : #같은업무
    k=0
  elif df1_outerKey == df2_outerKey and df1_innerKey == df2_innerKey: #상위하위 혹은
비슷한업무
    k=1
  elif df1_outerKey == df2_outerKey : #같은계열업무
    k=2
```

```
    elif df1_outerKey != df2_outerKey :
        k=3 #다른계열업무
    return alpha ** k

#경력 거리 계산
def cal_career(df1, df2, alpha = 0.7):
    career_dict={1:['대신증권', '대신증권 WM 사업단장', 'WM 추진부문장', '재경 1WM 부문장',
'대신증권중부지역본부장', '기획본부장대신증권', '대신증권분당지점장', '대신증권무거동지점장',
'대신증권 Capital',
                    '이사대우부본부장', '기획본부장대신증권인재역량센터장',
'대신증권인재전략부', '대신증권파생상품운용부장', '대신증권자산운용본부',
'대신증권영업기획부장', '대신증권기획실장',
                    '대신증권홍보실', '대신증권브랜드전략실', '대신증권 IT 개발부장',
'대신증권트레이딩시스템부장', '대신증권 Global 사업본부', '신증권', '기업금융사업단장'],
#대신증권
                0:['대신에프앤아이', '대신투자신탁운용', '대신투자신탁운용상무이사',
'대신에이엠씨', '대신자산운용', '대신저축은행'], #대신증권 자회사
                2:['메리츠종금증권', '메리츠증권', '미래에셋증권', 'KTB 투자증권',
'SC 제일은행', '하나금융투자', 'IBK 투자증권', 'DB 금융투자', 'NH 투자증권', 'KB 증권',
'LIG 손해보험', '굿모닝신한증권', '메릴린치증권',
                    '삼성증권', '동양증권', '한국투자증권', '한국투자증권평촌지점', '대우증권',
'대우본부장', '대우증권전략기획본부', '대우증권해외사업부문', '대우증권 IB 사업부문',
                    '우리투자증권', '우리 CS 자산운용', '우리투자증권기업금융 2 팀장',
'우리프라이빗에쿼티', '푸르덴셜투자증권', '하이투자증권주식인수팀',
'하이투자증권주식인수팀', 'SBC', 'Bank', '도이치은행'], #타증권사
                3:['금융감독원', '금융위원회', '現)금융위원회', '감사원', '대검찰청', '법무법인',
'現)법무법인', '세무법인', '국세청', '중부지방국세청', '서울북부지방검찰청','서울지방국세청장',
'서울지방국세청',
                    '제 58 대', '제 22 대', '기획재정부', '국세청,관세청,산업통상자원부',
'자본시장연구원', '한국조세연구원', '국가청렴위원회', '대한중재인협회', '한국회계정보학회',
                    '금융위원회금융발전심의위원(현)', '금융위원회적극행정심의위원(現)',
'안진회계법인',  'L&C 세무회계사무소', '변호사정상명법률사무소', '피앤비세무컨설팅',
'現)법무법인(유)'], #법,정책
                4:['중앙대', '중앙대학교', '現)중앙대', '연세대학교', 'KAIST', '수원대',
'수원대학교', '서울시립대', '現)서울시립대', '학교법인', '現)학교법인', '재단법인', '現)재단법인'],
#학교, 교수, 재단
                5:['하비스트', '한국물산', '우리선물', '부영주택', 'LG 경제연구원', 'Arthur']
#기타

                }
```

```
  #경력 여부 빈리스트 생성
  df1_list = [0,0,0,0,0,0] #0 번인덱스: 대신증권 자회사, 1 번인덱스: 대신증권, 2 번인덱스: 타증권사, 3 번인덱스: 기타
  df2_list = [0,0,0,0,0,0]

  #리스트에 각 업종에 종사했었으면 1, 아니면 0
  for career in df1['주요경력요약'] :
    for key, val in career_dict.items() :
      if career in val :
        df1_list[key]=1

  for career in df2['주요경력요약'] :
    for key, val in career_dict.items() :
      if career in val :
        df2_list[key]=1

  #df1_list 와 df2_list 합 구하기(요소가 0-> 둘다 종사 안했음, 1-> 한쪽만 종사함, 2-> 둘다 종사함)
  sum_list = [x + y for x, y in zip(df1_list, df2_list)]

  #k 값 구하기
  if df1_list == df2_list : #전체 같으면 0
    k=0

  elif (sum_list[0]==0 and sum_list[1]==0 and sum_list[2]==0): #기타만 있을 때 (법률, 학교, 기타)
    count2 = sum_list.count(2) #2 인 요소 개수 찾기(df1, df2 같은거 개수)
    k = 0.6 * (1 + count2)

  else : #기타만 있는거 아니고,
    #법률, 학교, 기타 전부 기타로 통합. 범주는 3 으로.
    if sum(df1_list[3:]) > 0 :
      df1_list[3] = 1
      df1_list[4] = 0
      df1_list[5] = 0
    if sum(df2_list[3:]) > 0 :
      df2_list[3] = 1
      df2_list[4] = 0
      df2_list[5] = 0
```

```
    #수정한 df1_list, df2_list 로 sum_list 다시 생성
    sum_list = [x + y for x, y in zip(df1_list, df2_list)]

    #df1_list, df2_list 에서 1 인 인덱스 뽑기
    index1 = [i for i in range(len(df1_list)) if df1_list[i] == 1]
    index2 = [i for i in range(len(df2_list)) if df2_list[i] == 1]

    min = 10
    max = 0

    for i in index1 :
      for j in index2 :
        if i != j: #두 사람 간에, 다른 업종끼리 최대거리 최소 거리 구하기

          if (i==3 and j != 3) or (j==3 and i != 3) : #기타와 다른 업종
            max = 4 #기타는 모두와 거리 4 고정 (최대거리 구하기)
            temp = 4
          else : #최대거리 구하기
            temp = abs(i-j)
            if temp > max:
              max = temp
          if temp < min: #최소거리 구하기
            min = temp

    k=float((max+min)/2) #거리는 최대거리, 최소거리의 평균
    k= k- (0.5 * sum_list.count(2)) # 같은 업종 있으면, 있는 만큼 거리 -0.5

  return alpha ** k

#보유 주식수별 거리 계산
def cal_stock(df1, df2, alpha = 0.7):
  #k 값 구하기
  if df1['의결권 있는 주식수'] == df2['의결권 있는 주식수'] :
    k=0
  else :
    k= abs(df1['의결권 있는 주식수'] - df2['의결권 있는 주식수'])//100 #차이로 보기=>
차이가 거리가 됨(너무 차이 크면 거리 멀어짐), 100 개 단위로 끊어서 보기
    k=math.log10(1+k) #로그스케일
  return alpha ** k
```

```
# In[5]:


#가중치 함수
def weight_sum(df1, df2):
    wt_college = cal_college(df1, df2)
    wt_department = cal_department(df1, df2)
    wt_position = cal_position(df1, df2)
    wt_task = cal_task(df1, df2)
    wt_work_years = cal_work_years(df1, df2)
    wt_career = cal_career(df1, df2)
    wt_stock = cal_stock(df1, df2)
    return wt_college + wt_department + wt_position + wt_task + wt_work_years + wt_career
+ wt_stock


# In[6]:


# 인물 가중치행렬
col = []
row = []
matrixs = []
dataFrames = [df_2014, df_2015, df_2016, df_2017, df_2018, df_2019, df_2020, df_2021,
df_2022, df_2023]

for dataframe in dataFrames:
    for i in range(len(dataframe)):
            col.append(dataframe.loc[i, '이름'])
            row.append(dataframe.loc[i, '이름'])
    weight_matrix = pd.DataFrame(columns=col, index=row)
    matrixs.append(weight_matrix)
    col=[]
    row=[]


# In[9]:
```

```
#가중치 계산
for t, dataframe in enumerate(dataFrames):
    for i in range(0, len(dataframe)):
        for j in range(0, len(dataframe)):
            matrixs[t].iloc[i, j] = round(weight_sum(dataframe.iloc[i], dataframe.iloc[j])**2, 2)


# In[ ]:


matrixs[0]


# ### 중요인물선정

# In[13]:


# 대신증권 연간 재무 정보
daishin_Annual_Financial_Info = pd.read_csv('data/daishin_Annual_Financial_Info.csv', sep=",",
encoding='UTF-8')
profit_avg = daishin_Annual_Financial_Info['연도별 누적 순이익 (단위:천원)'].mean()

def find_promoted_employees(df, daishin_Annual_Financial_Info):
    increased_years = []  # 매출이 증가한 연도를 담을 리스트
    promoted_employees = []  # 승진 또는 입사한 인물을 담을 리스트

    # 매출액이 증가한 연도 찾기
    for index, row in daishin_Annual_Financial_Info.iterrows():
        if row['연도별 누적 순이익 (단위:천원)'] > profit_avg:
            increased_years.append(row['연도'])

    for year in increased_years:
        for index, row in df.iterrows():
            if row['연도'] == year:
                previous_year = year - 1

                previous_row = df.loc[df['연도'] == previous_year]
                start_year = int(row['재직기간'].split('.')[0])
```

```
                    # 입사 여부 확인
                    if start_year == year:
                        promoted_employees.append(row['이름'])

                    # 직위 변화 확인
                    elif len(previous_row) > 0 and row['직위'] !=
previous_row['직위'].values[0]:
                        promoted_employees.append(row['이름'])

    return promoted_employees

promoted_employees = find_promoted_employees(df, daishin_Annual_Financial_Info)

def find_increased_stockholders(df, daishin_Annual_Financial_Info):
    increased_years = []  # 매출이 증가한 연도를 담을 리스트
    increased_stockholders = []  # 주식보유량이 증가한 인물을 담을 리스트

    # 매출액이 증가한 연도 찾기
    for index, row in daishin_Annual_Financial_Info.iterrows():
        if row['연도별 누적 순이익 (단위:천원)'] > profit_avg:
            increased_years.append(row['연도'])

    for year in increased_years:
        for index, row in df.iterrows():
            if row['연도'] == year:
                previous_year = year - 1

                # 주식보유량 변화 확인
                previous_row = df.loc[df['연도'] == previous_year]
                if len(previous_row) > 0 and row['의결권 있는 주식수'] >
previous_row['의결권 있는 주식수'].values[0]:
                    increased_stockholders.append(row['이름'])

    return increased_stockholders

increased_stockholders = find_increased_stockholders(df, daishin_Annual_Financial_Info)

important_person = promoted_employees + increased_stockholders

from collections import Counter
```

```
counter = Counter(important_person)

# 등장 횟수가 많은 순서대로 정렬된 튜플 리스트 생성
sorted_elements = sorted(counter.items(), key=lambda x: x[1], reverse=True)

# 정렬된 리스트에서 요소와 등장 횟수 출력
for element, count in sorted_elements:
    print(f'{element}: {count}')


# In[19]:


def select_important_people(sorted_elements, threshold=2):    # threshold 이상 등장한 인물을 리스트로 반환
    import_p = []
    for element, count in sorted_elements:
        if count >= threshold:
            import_p.append(element)
    return import_p


# In[76]:


major_p = select_important_people(sorted_elements, 2)
print(major_p)


# ### 네트워크 시각화

# In[79]:


import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

def print_network(matrixs):
```

```
    for k in range(len(matrixs)):
        node = matrixs[k].columns.tolist()
        # 특정 임계값보다 작은 가중치를 가진 간선은 표시하지 않음 (centrality 계산을 위해 다시 아래 코드를 삽입한것임)
        m_w_sum = []
        for i in range(len(matrixs[k])):
            for j in range(len(matrixs[k])):
                m_w_sum.append(matrixs[k].iloc[i, j])

        threshold = np.percentile(m_w_sum, 90)
        node = matrixs[k].columns.tolist()
        # 그래프에 노드와 간선 삽입
        G = nx.Graph()
        G.add_nodes_from(node)
        for i in range(len(node)):
            for j in range(i + 1, len(node)):
                weight = matrixs[k].iloc[i, j]
                if weight >= threshold:  # 특정 임계값 이상인 경우에만 add_edge
                    G.add_edge(node[i], node[j], weight=matrixs[k].iloc[i,j])
        plt.figure(figsize=(12, 20))
        plt.rcParams['font.family'] = 'Malgun Gothic'
        pos = nx.spring_layout(G, k=3.0)  # 레이아웃 설정
        node_colors = ['red' if n in major_p else 'lightblue' for n in G.nodes()]  # major_p 에 해당하는 노드는 빨간색, 나머지는 파란색으로 설정
        weights = [G[u][v]['weight'] for u, v in G.edges()]  # 엣지의 가중치 리스트

        degree_centrality = nx.degree_centrality(G)
        node_sizes = [250 if degree_centrality[n] * 10000 <= 50 else degree_centrality[n] * 15000 for n in G.nodes()]    # degree centrality 에 따라 노드 크기 조정

        nx.draw_networkx(G, pos, with_labels=True, node_color=node_colors, node_size=node_sizes, edge_color='gray', width=1.0, font_family='Malgun Gothic')

        # 가중치에 따른 엣지 라벨 추가
        # labels = nx.get_edge_attributes(G, 'weight')
        # nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
가중치를 표시하려면 주석 처리 x
```

```
        # Create legend handles and labels
        legend_handles = [
            mpatches.Patch(color='red', label='중요인물'),
            mpatches.Patch(color='lightblue', label='일반인물')
        ]
        plt.legend(handles=legend_handles)
        plt.title('{0}년도 네트워크'.format(2014+k), color='blue', fontsize=16)
        plt.show()

        # 각 인물들이 중요 인물과 얼마나 연결되어있는지 출력
        sorted_nodes = sorted(G.nodes(), key=lambda n: sum(1 for neighbor in
G.neighbors(n) if neighbor in major_p), reverse=True)
        for n in sorted_nodes:
            connections = sum(1 for neighbor in G.neighbors(n) if neighbor in major_p)
            print(f"{n}은 중요인물과 {connections}개 연결됨")

        # 중심성 계산
        degree_centrality = nx.degree_centrality(G)
        betweenness_centrality = nx.betweenness_centrality(G)
        closeness_centrality = nx.closeness_centrality(G)
        eigenvector_centrality = nx.eigenvector_centrality(G)

        # 결과 출력
        print("{0}년도 네트워크 Centrality 계산".format(2014+k))
        print()
        print("Degree Centrality:")
        for node, centrality in sorted(degree_centrality.items(), key=lambda x: x[1],
reverse=True):
            print(node, centrality)
        print()
        print("Betweenness Centrality:")
        for node, centrality in sorted(betweenness_centrality.items(), key=lambda x: x[1],
reverse=True):
            print(node, centrality)
        print()
        print("Closeness Centrality:")
        for node, centrality in sorted(closeness_centrality.items(), key=lambda x: x[1],
reverse=True):
            print(node, centrality)
        print()
```

```
        print("Eigenvector Centrality:")
        for node, centrality in sorted(eigenvector_centrality.items(), key=lambda x: x[1],
reverse=True):
            print(node, centrality)
        print()


# In[80]:


print_network(matrixs)
```