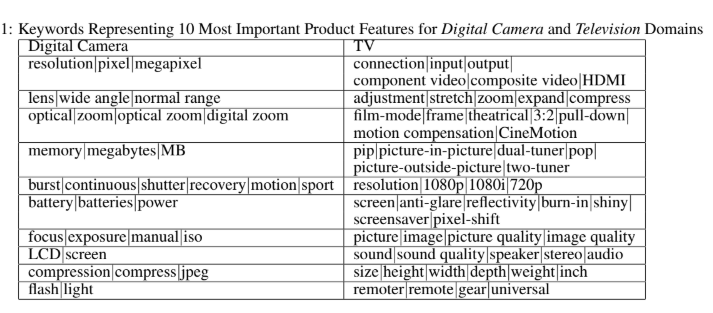
Voice of the Customers: Mining Online Customer Reviews for Product

* **논문 구현 순서**

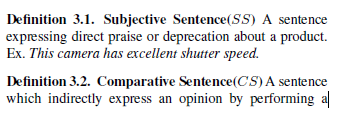
1. 제품군 별 제품 features와 리뷰들을 가져온다
2. 문장을 features별로 나눈다.(전처리)
3. 문장들을 4개로 구분한다; 긍정주관, 부정 주관, 긍정 비교, 부정 비교
4. 데이터프레임에 “문장”(feature , PS/NS, PC/NC)이런 식으로 저장
5. 문장의 sentiment 구분
6. 각 노드들을 연결하는 Graph를 만들고
7. 순위를 매긴다.
8. 순위 비교 한다. (RFF & IF)

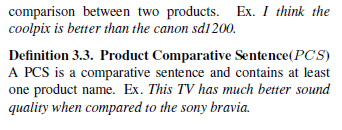
* **Indentifying product features**



1. 문장에 동의어/동의문장이 있으면 ‘describing feature’라고 구분한다.
2. 유의어들을 하나의 단어로 합친다.

* **Sentence labeling**





1. MxTerminator[11] to split reviews into sentences because a typical customer review comprises of several sentences.
2. Feature 단어가 나타나는 문장을 추출하고 추출된 문장을 labeling한다.(NC/PC, NS/PS)

* **Identifying comparative sentences**

순서: KW사전 확보 -> 비교할 때 쓰이는 단어의 POS 태그 확보 -> 특정 패턴 정의(as <word> as, the same as, silmilar to) -> 문장에는 제품 이름이 2개 이상 나와야 비교 문장으로 인식. 어떻게 제품 이름 구별하느냐? 자연어처리, 개체명인식 NER을 통해서 태그로 구별할 수 있을 듯함.

1. KW contains 126 words, some of which are explicit(“outperform, exceed, compare, superior, etc.”) and others are implicit(“prefer, choose, like, etc.”)), sentence semantics, and sentence structure to identify comparative sentences.

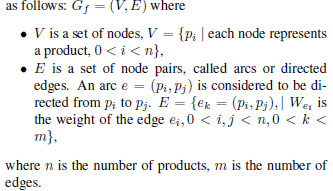
* 비교 문장 구별에 쓰이는 사전들:
* KW사전
* POS tags: JJR, RBR, JJS, RBS.
* Predefined structural pattern.(as <word> as, same as, similar to)

1. 단어들의 품사 태그 중 JJR, RBR, JJS, RBS를 구별한다
2. 문장 내 사전 정의된 패턴이 있는지 확인.
3. 문제점: “I bought this camera for my son because he got a higher grade in his second statistical exam.” 다른 제품과 비교하는 문장이 아님에도 higher 때문에 선택될 수 있음. 제품과 제품간의 비교 문장만 가져오기 위해 논문에서는
4. 해결: We use only comparative sentences which contain at least one product name which is different from the product the sentence is describing while building our ranking model. 비교 문장 안에 describing product 제외하고 **다른 제품의 이름이** 나오면 ‘비교 문장’으로 인식.

* **Identifying sentence sentiment orientation(극성 분류)**

1. we use a simple yet powerful method by utilizing a positive word set(POS) and a negative word set(NEG) developed in the MQPA project[19]. 이 외에 필요 단어들을 커스텀 사전으로 구축.
2. At the end of this process, we get a list of **1974** words for the positive set and **4605** words for the negative set4
3. If the sentence contains a word that is in the positive word set, we label this sentence with a positive tag. Negative sentiment words are handled similarly.
4. We constructed a set of 28 negation words manually. – 커스텀 사전

* **Contstructing the Product Graph**



* V = f특성을 가진 product의 총 sentence 수, 노드의 weight값을 구하는 식이 들어간다.
* Gf = (V, E) Gf를 V와 E에 대한 함수로 정의할 수 있다.
* E = 연결된 노드들의 수
* Pi = i번째 프로덕트
* 0 < i < n 프로덕트의 갯수
* E = 밑 그림 속 라인 별로 퍼지는, edge의 weight 값.

**Graph 만드는 순서**

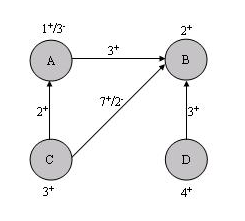
1. Use pRank Algorithm
2. pRank인 이유는 classic PageRank는 모든 edge에 동일한 가중치를 두고 node weight는 사용하지 않는다. 그러나 pRank는 가중치를 줌으로써 제품들 간 상대적인 중요도(edge weight) 뿐만 아니라 해당 제품 그 자체의 quality(node weight)까지 고려할 수 있게 된다. 그래서 edge weight만큼이나 **node weight가 중요하다!**
3. i번째 P의 f라는 feature에 대한 비교 리뷰, Pi를 Pj와 비교할 때 Pj에서 Pi로 edge를 추가한다.
4. Assign a weight to this edge. 엣지에 가중치를 줌.

* **Edge Weight 구하는 방법**

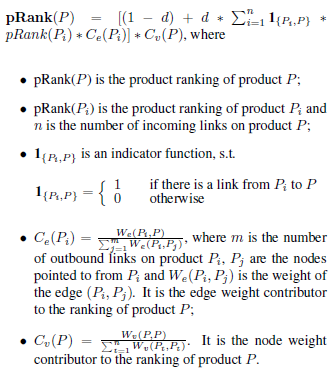
1. The review = a positive comparative(PC(Pi; Pj)) IF Pi 가 Pj보다 좋으면.
2. The review = a negative comparative(NC(Pi; Pj)).IF Pi 가 Pj보다 안좋으면.
3. 각 엣지마다 PC와 NC 문장들을 count한다.
4. PC를 NC로 나누어 해당 엣지(Pj에서 Pi로 가는 엣지)의 가중치를 구할 수 있다. =PC/NC
5. 해당 가중치는 product의 feature의 quality를 나타낸다.

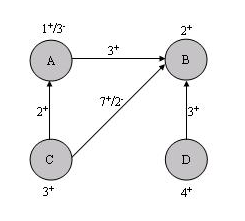
* **Node Weight 구하는 방법**

1. Node weight 는 product 자체의 quality를 나타낸다. = PS/NS

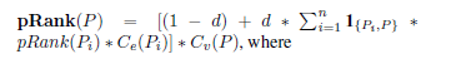


* **Ranking Products**





원래 방향이 일방이라는 점이 걸려서 B -> C로 갈때의 가중치와 C -> B로 갈때의 가중치가 서로 달라진다는 점을 고려해서 계산을 해보려했는데 굳이 그럴 필요가 있나 싶었다. 랭킹 순위를 매기는게 이 모델의 목적. C -> B이든 B -> C이든 C기준에서 B보다 C가 좋은거고 B 기준에서 B보다 C가 좋은건 달라지지 않는다. 순위는 어느 방향이든 B가 C보다 낮기 때문에 두 방향의 edge weight를 구하는 부분에 대한 의문이 들었다.



P의 순위를 구하는 공식 = ~~~P와 연결된 노드들의 순위

pRank(Pi) = 1로 대입

이거는 P와 Pi가 연결되어있을 때 값을 1 return해라. 그렇지 않으면 0.시그마는 대괄호 닫히는 부분까지 연결되어 있음



Wv(P, P) = 자기 자신에 대한 edge 가중치. 즉 P가 스스로와 연결되어 생기는 edge의 값.

Node\_weight(P) = edge\_weight(P)

Wv(Pt, Pt) = 모든 노드 자신들의 가중치의 합

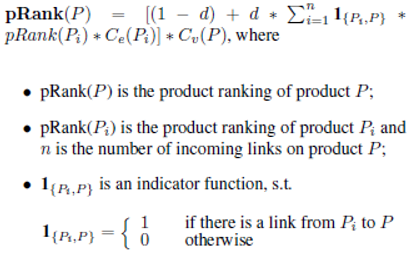
자기자신 / 전체

---------------------



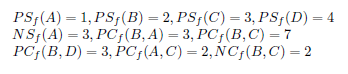
들어오는 엣지들의 노드들의 가중치

----------------

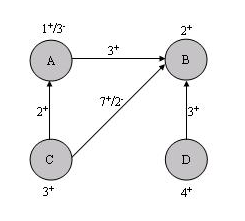


위 식에서 시그마가 ...\*Ce(Pi)까지 연결되어있다. i= 1부터 n까지 반복할 때 연결이 되지 않는 Pi값들은 0으로 두고 계산한다. 1{Pi, P}덕분에 P와 연결되지 않은 노드들을 분리하여 계산할 수 있게됨.

**\*\*\*\*예문을 통한 논문 사용법:**

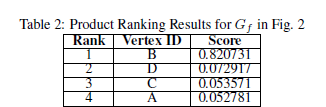
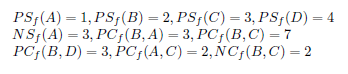


1. 제품 A, B, C, D를 f라는 특성으로 순위를 매기려고, 제품별 특성 f에 대한 Positive/Negative, Subjective/Comparative의 개수를 센다.

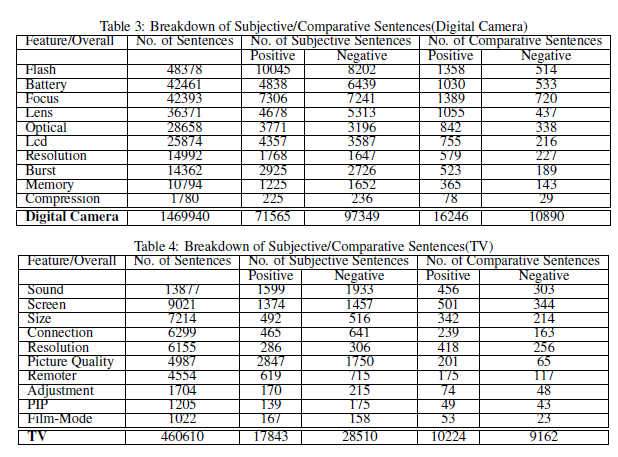


1. **Edge weights** are determined by comparative sentences, and **node weights** are determined by subjective sentences. Edge weights = PC/NC, node weights = PS/NS
2. It must be mentioned that to prevent edges with infinite length (when the number of negative comparative sentences is 0), we set the minimum value of the **denominator to 1** 분모의 최소값을 1로 둔다.
3. 값이 적을 수록 좋은 제품이다. The ranking order(the smaller, the product

better) for this graph is B ! D ! C ! A

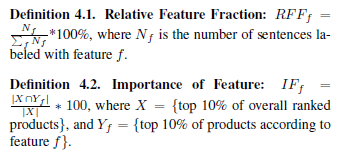


* **Experiment Results**



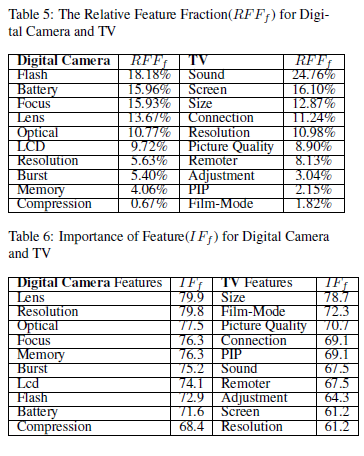
To evaluate our ranking algorithm, we first perform product ranking based on the overall quality. To determine the overall rank of a product, we include all comparative and subjective sentences in our database while constructing the product graph. \*\*\* **Overall rank를 구해야함 아래 IF에서 특정 feature f와의 계산에서 overall rank가 쓰임.**

* In this paper, we focus on the feature-specific ranking obtained by mining the individual product graphs generated for each product feature 제품마다의 각 특성의 graph를 구함.
* **어떤 feature가 customer에게 중요한가?**



**RFF**는 특징 f가 들어간 문장이 전체 특성 문장들 중 얼만큼의 비중을 차지하는지 구함. RFF = (특정 f가 들어간 문장 수)/(모든 feature의 문장 수)

IF는 overall rank와 feature specific rank간의 유사도를 보고 제품을 고를 때 소비자들이 생각하는 중요한 특성이 무엇인지 파악할 수 있음.



RFF = 전체 문장들 중에서 특정 feature의 빈도수를 측정함

IF = 전체 문장의 탑 10%와 특정 feature의 탑 10% 교집합 / 전체문장의 탑 10%. 전체 순위와 특정 feature 순위를 비교하여 그 feature의 중요도를 파악.