1. 3/16일 수업 내용

Q1: EUC-KR같은 경우 1바이트의 아스키 코드 영역과 2바이트의 섞여있는데 그러면 EUC-KR도 유니코드와 같은 가변 너비 인코딩의 일종인지 궁금합니다.

A1: 7)번 문제와 관련이 많으니 다른 학생들도 잘 봐주세요. EUC-KR = KS완성형 + ASCII 입니다. EUC-KR이나 ASCII 등은 문자들의 집합입니다. KS완성형에 한글코드는 2350자가 정의되어 있구요.(참고로 한자 4,888자, 자음/모음/문장부호 등이 94x94 영역에 정의됨)

즉, EUC-KR은 ASCII에 정의된 문자들과 KS완성형에 정의된 문자들의 "합집합"입니다. EUC-KR은 두 문자셋의 합집합인데 독자적인 이름이 필요한 이유는 "한글 리눅스"에 지원하는 "한국사람들이 사용하는 linux(또는 unix) 운영체제"에서 사용할 수 있는 "텍스트 문자셋"을 명확하게 정의하고 naming을 해야할 필요가 있었기 때문입니다.

Q2: KS 완성형 한글코드는 11,172자 중 자주 사용하는 2,350자만 코드 부여를 하는데, 이 우선순위는 어떻게 정해진 것인가요? 이 또한 빅데이터 분석으로 정해진 것인가요?

A2: 일종의 빅데이터 분석이라고 할 수 있겠네요. KS완성형을 정의하던 당시에는 데이터가 많지 않아서 빅데이터까지는 아니지만... 신문/잡지/문학작품 등 한글문서들에 대한 사용빈도 조사에 의해 빈도가 높은 2,350개를 선별하였습니다

2. 3/18일 수업 내용

Q1: 과제 1의 <방법 공통 음절 개수, 공통 어절 개수, 공통 어휘 개수 등에 의한 유사도 계산>에서 어휘와 어절의 차이는 무엇인가요?

A1: "꽃이 피었다"를 예로 들면 아래와 같습니다.
- 어절: '꽃이', '피었다'
- 어휘: '꽃', '피다'

Q2: utf-8 인코딩 방식에 BOM이라는 형식(?)도 있는 데 이는 어느 상황에서 사용되나요?

A2: 봄은(가을 아니고 ^^)...
윈도에서 4가지 문자셋으로 텍스트 파일을 저장할 수 있으므로 .txt 파일이 어떤 문자셋으로 저장되었는지를 표시하기 위한 제어문자입니다. 윈도 "메모장"에서 .txt 작성할 때 파일의 첫 2개 또는 3개 문자로 삽입됩니다.
0xFEFF, 0xFFFE, 또는 0xFEFF를 utf8로 인코딩한 3바이트

3. 3/23일 수업 내용

Q1: 어절의 경우 문장부호를 별개의 어절로 간주하라고 하셨는데
1. 촉촉한 촉촉한 초코칩이다.
2. 안 촉촉한 촉촉한 초코칩이 아니다.
라고 가정했을 때 1번 문장의 어절은 마침표 포함해서 4개의 어절인지 궁금합니다.
그리고 마침표도 공통 어절로 포함시켜야 하는 건지 궁금합니다.

A1: 마침표 등 문장부호를 유사도 계산에 포함할지의 문제는... 유사도 계산을 어떤 목적으로 활용할지에 따라 다르므로 옵션값에 의해 두 가지를 모두 가능하도록 구현하는 것이 바람직합니다
수업의 과제에서는 실용적인 목적이 아니고, 연습을 위한 목적이므로 하고싶은 대로 하면 되고, 결과 출력할 때는 문장부호 없는 입력으로 테스트해도 됩니다.

Q2: 실습 영상의 kma파일을 보면 결과에서
최종라운드서
(N : 최종라운드서)< :60)
(N : 최종라운드)< :70) + (j '에서')<4>
라고 나와있는데 이런식으로 두개씩 나오는 이유가 사전에 없어서 두가지의 분석으로 추정했다는 건가요?
그러면 :60은 사전에 없는 단어라고 말씀하셨는데 <:70>이나 <4>는 무슨 뜻인가요?

또 색인분석 영상에 나온 올스타전과 같은 복합명사 분해를 잘못한 경우는 ham-cnn.dic에 규칙을 넣어주신다 하셨는데 규칙을 어떻게 넣는건가요?

A2: 콜론 앞에 영문자로 품사 표시를 하는데요. 영문자가 없으면 사전에 없는 어휘입니다. 콜론 뒤의 숫자는 미등록어 유형 구분을 위한 것으로 형태소 분석기의 내부 정보를 출력한 것이니 무시하면 됩니다. 조사 '에서' 뒤의 4라는 숫자도 마찬가지인데, 이 경우는 '서'로부터 '에서'로 '에' 복원 표시입니다.

Q3: Dictionary에서 기분석 사전에 규칙이 없는 특별한 형태의 분석을 원할 때 에디팅 해주신다고 하셨는데, 적절한 예를 알 수 있을까요??

A3: hdic/ham-rma.dic 살펴보면 이미 등록해 놓은 여러가지 예제들이 있습니다. 아래...
개운치 <개운, N> + <하, t> + <지, e>
거니 <것, U> + <이, c> + <니, e>
거다 <것, U> + <이, c> + <다, e>
거란 <것, U> + <이, c> + <란, e>
거로군요 <것, U> + <이, c> + <로군요, e>

건 <것, U> + <은, j>
걸 <것, U> + <을, j>

Q4: 합성어의 경우, 의도한 것과는 다르게 분해되는 것들을 몇몇 보았는데, 올바르게 고치는 데에 어떤 방법이 있을까요?

A4: 합성어'의 의미는 '복합명사'겠지요? 복합명사는 명사+명사 패턴이므로 각 '명사'들이 사전에 있는지 등의 정보를 이용하여 자동 분해를 합니다. 자동 분해가 100% 완벽할 수는 없으므로(분해 규칙에 맞지 않는 경우 발생함) 이런 경우에는 직접 복합명사를 ham-cnn.dic에 분해방법을 지정해 주면 그 정보대로 분해를 합니다.

만약에 시스템 사전(hangul.dic)에 등록된 명사라면 단위명사(복합명사가 아니라)이므로 분해를 하지 않습니다. '호랑나비'의 경우가 여기에 해당됩니다. 이 경우에 '호랑'+'나비'로 분해해야 한다면 hangul.dic에 등록된 '호랑나비'를 삭제(미등록으로 처리)해야 하는데 hangul.dic은 binary 형태로 구성되어 있기 때문에 사전관리 프로그램을 통해서 삭제처리를 해야 합니다. 조금 복잡한 문제네요.

'호랑나비'를 꼭 분해해야 한다면 형태소 모든 분석결과들에 대해서 '호랑나비'이면 '호랑'+'나비'로 분해하도록 후처리에서 처리하는 게 쉽고 편한 방법입니다.

index.exe 실행할 때 "C> index -2"와 같이 "-2" 옵션을 주어 실행하면 추출된 명사가 사전에 있는지 'N', 사전에 없는 것인지 'K'로 구분하는 정보를 출력해 줍니다. '호랑나비'는 'N' 표시이므로 hangul.dic에 등록된 명사라는 것을 알 수 있습니다.
참고로, 'P' 표시는 복합명사를 분해했을 때 단위명사들의 표시입니다.

4. 3/25일 수업 내용

Q1: 검색엔진들은 대부분 ngram 기법을 사용해 색인하나요? 사용한다면 같이 사용 되는 기법은 무엇이 있나요?

A1: Lucene 등 다국어 검색엔진은 DBCS 언어(한중일)를 위해 기본적으로 ngram 색인기법을 제공합니다.
영어처럼 1개의 형태소가 단어를 구성하는 언어는 굳이 ngram 기법을 사용할 필요가 없고, 대신에 stemmer 또는 tokenizer를 사용합니다.

Q2: 실제로 유사도를 측정할 때는 빈도수로 하는 것인지 궁금합니다.
혼자 생각을 해보았을 때 비슷한 두 문장이 있는데,
1. 내가 그린 기린 그림 / 기린이 그린 내 그림 의 유사도는 빈도수로 보았을 때는 높은데 의미는 완전히 다르다고 생각이 듭니다. 이럴때도 빈도수로 카운팅을 하여 유사도를 보는지 궁금합니

다.
2. 위와 같은 경우에는 유사도를 어떤식으로 검사하는지 다른 방법이 있는지 궁금합니다.

A2: ngram 기법 이외에 다른 유사도 계산 방법을 사용할 수도 있습니다.
어떤 방법을 사용할지는 여러가지 요인을 고려하여 사용자가 결정할 문제입니다.

Q3: 네이버 검색창에 '했급니다'를 입력했을 때 검색창 아래에 '했습니다'가 뜨는데 네이버의 검색엔진에도 ngram 기법이 사용되는 건가요?

A4: 네이버 등 포털에서 오류수정 추천어 생성 방식은 "주로" 사용자들이 검색어로 입력했던 검색어들을 DB에 저장 및 빈도 카운트를 하여, "했급니다"가 입력되면 DB를 탐색합니다.
검색어가 저빈도인 경우에 자모 1~2개 유사 또는 발음이 유사한 것 등 유사도 높은 검색어들을 후보로 추천합니다. 추천한 후보를 선택하면 검색어에 대해 사용자가 선택한 것의 정보를 DB로 관리하여 다음에 유사도 추천할 때 활용합니다.

포털의 후보추천(오류교정?) 방식은 HWP나 MS word 등 워드프로세서의 spelling correction 방식과 다릅니다. 검색포털의 검색어는 기존의 다른 사용자들이 입력했던 엄청나게 많은 검색어 DB가 있으므로 이를 활용하여 검색 환경에서 고빈도인 것을 추천하면 사용자 만족도가 높습니다.
반면에 워드프로세서에서는 기존의 다른 사용자들의 정보를 활용하기 어렵기 때문에 이 방법을 활용할 수가 없지요.

Q4: 철자 오류가 발생했을 때 문맥에 따라 다르게 교정돼야 한다고 생각합니다. 따라서 해당 어절과 유사한 어절의 출현확률 뿐만 아니라 이전 어절과 문맥에 맞게 이어지는 건지도 고려하나요??

A4: 초기의 스펠러 개발 당시에는 "어절 bigram, trigram" 정보 등 좌우 문맥을 고려할 만큼 텍스트 정보를 확보하기가 어려웠습니다. 따라서 unigram 빈도에 따른 어절 자체의 출현 확률과 사람들이 주로 오류를 범하는 규칙에 의한 spelling correction을 했는데요. 좌우 어절을 고려하지 않으므로 잘못 교정하는 경우(교정후보의 1위 어절이 정답이 아닌)가 종종 발생합니다.

요즘은 대용량 말뭉치를 쉽게 확보할 수 있으므로 좌우 문맥을 고려하는 교정 방법을 사용합니다.