데이터 사이언티스트 및 AI Engineer 기술 면접 정리

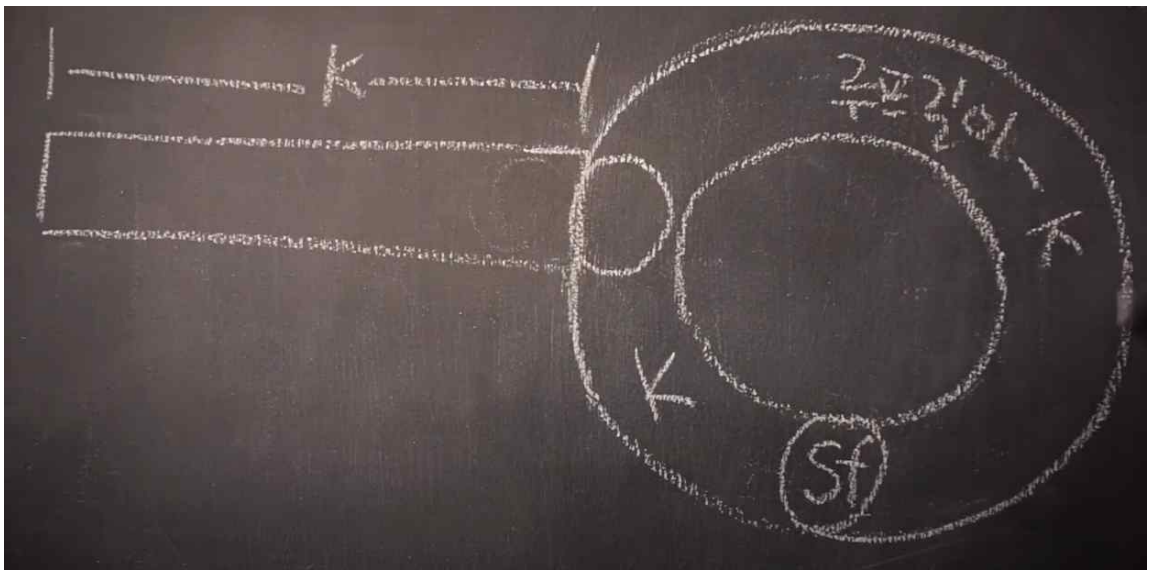
최현웅

1. 파이썬 관련 질문

1) [네카라쿠배] Linked List에서 순환 구조를 찾는 방법

여러 풀이법이 존재하겠지만, 2가지 방법이 떠오릅니다. 첫 번째로는 비효율적인 알고리즘으로써 BFS 알고리즘에서 방문 기록을 확인하는 부분을 약간 수정하여 순환 구조를 찾을 수 있다고 생각합니다. 방문 기록에는 해당 노드의 주소값을 저장하고 새롭게 들어오는 노드의 next의 주소 값과 방문 기록을 저장한 리스트에 해당 주소가 있으면 순환 구조라고 판단하는 방법으로써 이를 이용할 경우 방문 기록을 저장할 추가 공간이 더 필요하며, 해당 노드의 next와 방문 기록 리스트를 완전 탐색하여 보기 때문에 대략 O(n^2)의 시간 복잡도로 해결할 수 있을 것 같습니다.

가장 효율적인 풀이 방법으로는 이를 토끼와 거북이 알고리즘이라고 하는데, 토끼에 해당하는 노드는 한 번에 2칸씩 전진하고, 거북이에 해당하는 노드는 한 번에 1칸씩 전진하여 서로가 만날 때까지 반복문을 수행합니다. 해당 연결 리스트가 순환 구조라고 하였을 때 무한 루프가 되므로, 이 둘은 어느 순간에 반드시 만나게 됩니다. 서로 만나는 지점을 발견했을 때, 다시 거북이에 해당하는 노드를 head로 보내서 이번에는 토끼와 같은 속도로 탐색을 시작합니다. 당연히 토끼 노드는 제자리에서 다시 돌게 되니 순환 구조 안에서만 돌게 되는 특징이 있습니다. 사실 순환 구조에서의 이동 횟수는 k % loop length이고 첫 순환 노드의 주소는 그 지점에서 loop length - k를 이동하면 첫 노드를 찾을 수 있습니다. 이런 특징을 이용하여 두 개의 노드가 다시 만나는 지점을 순환 구조의 첫 노드라고 할 수 있습니다.

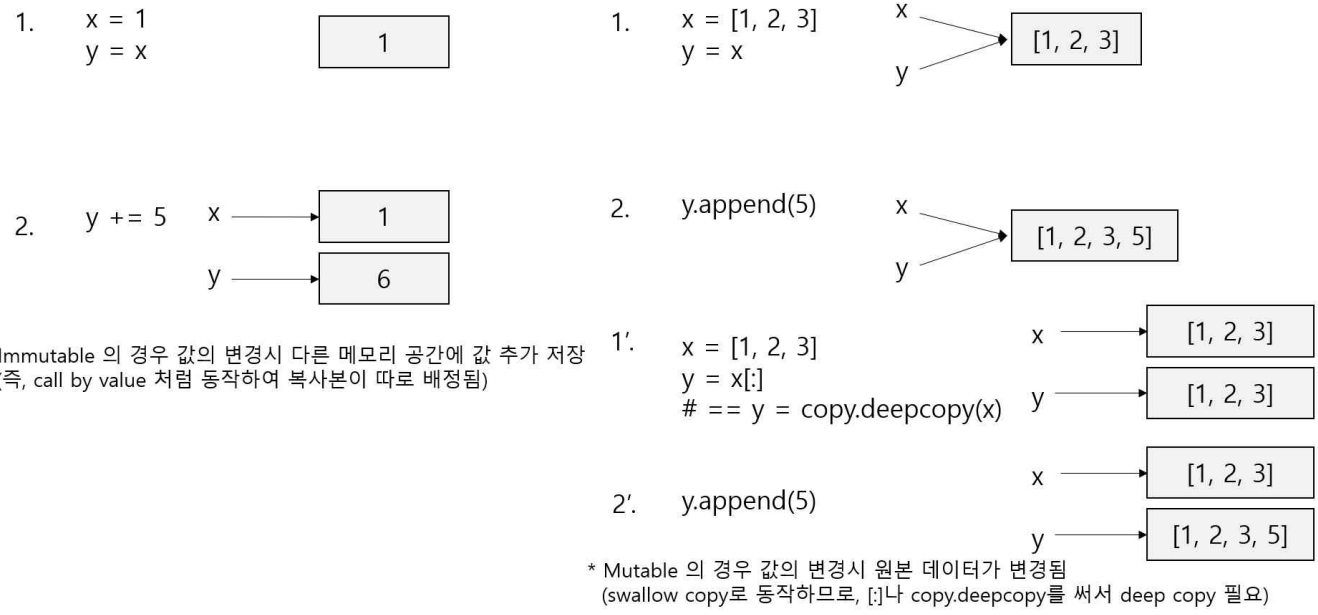


2) [네카라쿠배] Binary Search 사용시 최악의 경우의 예와 대처 방법

이진 탐색은 정렬이 되어 있는 배열에서 찾으려고 하는 값을 기준으로 현재 값과 비교하여 자료를 계속해서 반으로 나누며 탐색하는 방법입니다. 그렇기 때문에 사전에 정렬을 수행해주어야 하며, 이 경우에는 파이썬 내장 모듈인 sorted를 사용하는 경우 O(n logn)이 소요된다는 특징이 있습니다. 이진 탐색의 최악의 경우에 대해 계산해보면 데이터가 n개 일 때 n이 1이 되기까지 2로 나눈 횟수를 k회라 했을 때 이에 대한 시간 복잡도 함수는 T(n) = k + 1 (1은 마지막 원소의 비교)이며 이를 정리하면 T(n) = log_2 n이 됩니다. 즉 이진 탐색은 최악의 경우에도 O(logn)이며 이진 탐색을 쓸 때의 가장 최악의 경우에는 정렬 알고리즘을 이용하여 정렬할 때에 worst case가 나올 수 있습니다. quick & merge를 병합한 algorithm을 사용하여 최악의 경우에서도 정렬이 잘 수행되는 알고리즘을 채택하면 이런 문제를 해결할 수 있습니다.

3) [대기업, 네카라쿠배] 파이썬 Mutable과 Immutable의 차이

Mutable은 값이 변한다는 의미이며, Immutable은 값이 변하지 않는다는 의미입니다. x = 1이라고 선언하면 x라는 변수가 메모리에 1이라고 저장한 공간을 가리키는 일종의 레퍼런스 형태가 되는데, 여기서 y라는 변수가 y = x를 하면 x, y가 같은 공간을 가리키고, y += 10이라고 하였을 때, 파이썬에서는 11이라는 공간을 새롭게 할당하여 y 변수만 가리키는 주소를 바꾼다. 즉, Immutable의 경우에는 변수의 값이 변경되지 않으며, Mutable은 list, dictionary로써 하나의 레퍼런스에서 값을 변경하면 원본이 변경이 된다. 이를 막기 위해서는 [:]나 copy.deepcopy를 이용하여 깊은 복사를 해야한다. (call by value냐 call by reference냐의 뉘앙스)



4) [네카라쿠배] 파이썬 Map에서 Value로 정렬하는 방법

파이썬에서 sorted()라는 빌트-인 메소드를 이용하여 map를 정렬할 때 lambda function을 이용하여 map의 정렬 기준을 만들어 줄 수 있습니다. lambda 식에서 튜플의 0번째 index는 key가 되며 1번째 index는 value가 됩니다. 이런 특징을 이용하여 sorted() 메소드의 key에 lambda x:x[1]를 넣어주면 정렬을 수행할 수 있습니다. 하지만, sorted()의 매개변수로 map을 그대로 넣으면 list 형태로 변경이 되고, map 형태로 계속해서 유지하고 싶으면 map.items()를 매개변수로 넣어주어야 합니다.

5) [대기업, 네카라쿠배] 파이썬 List Comprehension

리스트 컴프리헨션은 말 그대로 리스트를 쉽게, 짧게 한 줄로 축약해서 만들 수 있는 기법입니다. 이에 대한 문법은 [(변수를 이용한 값) for (사용할 반복 변수) in (iterable)]으로 이루어져 있습니다. 사실 일반적인 반복문을 사용하였을 때 보다 리스트 컴프리헨션을 이용하였을 때 실행 속도가 훨씬 더 빠른 이점이 있습니다. 이를 좀 더 분석해서 살펴보면 일반 반복문의 경우 반복문을 돌때마다 특정 함수를 호출하여 수행한다는 특징이 있고, 리스트 컴프리헨션의 경우, 리스트 컴프리헨션 구문 안에서 익명 함수를 하나 만들고, iterable과 generator를 이용하여 함수의 출력 값을 계산하여 복사하는 방법으로 구성하여 function call의 횟수를 비약적으로 낮춤으로 실행 속도를 더 빠르게 했다는 장점이 있습니다. 더 쉽게 말씀드리면, for문마다 변수 할당을 하지 않으므로 속도가 더 빠르며 리스트 컴프리헨션에 range 구문을 이용하면 iterable를 생성하고 메모리에 이를 적재하지 않고 매 스텝마다 계산해서 넣어주기 때문에 더 빠르고 효율적인 실행속도를 보장할 수 있습니다.

6) [네카라쿠배] 파이썬 list와 tuple의 차이

list의 경우에는, 자유롭게 값을 변경할 수 있으나 tuple의 경우에는 값의 변경이 불가능하다는 특징이 있습니다. 그 외의 인덱싱, 슬라이싱, 연산 등의 모든 기능은 둘 다 똑같은 룰이 적용이 됩니다.

7) [네카라쿠배] 파이썬 list 구현 방식, 구현 방식별 insert / get 시간 복잡도

리스트를 구현하는 방식은 똑같이 배열을 이용한 방법과 연결리스트를 이용한 방법이 있습니다. 배열을 이용하는 방법은 모든 요소를 연속된 메모리 공간에 저장하며 검색 속도가 빠르다는 이점이 있으나, 삽입과 삭제 알고리즘에서 연결 리스트에 비해 비효율적입니다. 예를 들면, 배열로 구현한 list에 5000만개의 element가 저장되어 있고, 여기서 첫 번째 index에 element를 삽입한다고 하였을 때 가장 먼저 뒤의 5000만개의 element들을 한 칸씩 뒤로 shift 시켜야 한다는 최고 단점이 있습니다. 이는 O(n)이 소요됩니다. 삭제의 경우에도 똑같이 삽입의 case와 동일하게 작동합니다. 하지만 연결리스트로 list를 구현한다고 하면, 삽입과 삭제의 각 케이스에서 노드의 next만 적절하게 조정해주면 해결이 되므로 O(1)이 소요되나 탐색에서는 비연속적인 공간에 저장하므로 좀 더 느리게 동작합니다.

8) [네카라쿠배] 파이썬 dictionary 구현 방식, 구현 방식 insert / get 시간 복잡도

파이썬의 dictionary는 key와 value 형태로 이루어진 hash 기반의 자료구조라고 생각합니다. 이를 구현하기 위해 key는 항상 유일한 값이 될 수 있도록 해야하며, mutable한 객체를 key로 사용하면 해쉬충돌이 발생할 가능성이 크기 때문에 이를 key로 설정하는 것은 바람직하지 않습니다. dictionary에서의 insert, get의 연산은 key 값을 기반으로 바로 메모리 공간을 조회하기 때문에 모든 연산이 O(1)로 알려져 있습니다.

9) [네카라쿠배] 파이썬 dequeue 구현 방식, 구현 방식별 insert / get 시간 복잡도

dequeue는 양방향에서 자료의 삽입, 삭제가 가능한 알고리즘입니다. 이는 list와 연결리스트를 이용하여 구현할 수 있으며, list의 경우에 insert와 get의 시간 복잡도가 insert의 경우, 가장 맨 뒤에서 insert, get이 발생할 때에는 O(1)이 되지만, 맨 앞에서 이런 연산이 일어나는 경우 인덱스를 한 칸씩 다 이동시켜야 하므로 O(n)입니다. 이를 이중 연결리스트를 이용하여 구현하였을 때에는 모든 지점에서의 insert, get 연산이 O(1)로 될 수 있게 구성할 수 있습니다.

추가적으로 파이썬의 collections.deque는 기존의 list만을 사용하였을 때 보다 훨씬 더 빠른 실행속도를 보장할 수 있습니다. 예를 들어 pop 연산을 하였을 때 deque는 O(1)의 실행 속도이며, list는 O(n)의 실행 속도를 가지고 있습니다. 아마 내부적으로 연결 리스트와 해쉬를 이용하여 구성 했기 때문에 이런 실행 속도를 보장할 수 있는 것이라고 생각합니다.

10) [네카라쿠배] 파이썬에서 key가 될 수 있는 것은? list는 키가 될 수 있을까?

파이썬에서 key가 되려면 가장 먼저 해당 변수가 hashable 해야 합니다. 리스트 같은 mutable한 객체의 경우 unhashable 하기 때문에 이런 객체들은 키가 될 수 없습니다. 다만, immutable한 객체 중 튜플의 경우에는 hashable 하기 때문에 key 값으로 활용될 수 있습니다. 저 또한, 코딩테스트 문제를 풀면서 key 값의 기준이 여러 개가 될 때 key를 튜플로 구성해서 풀이했던 경험이 있습니다. 또한, 파이썬 딕셔너리에 대한 이야기를 좀 더 구체적으로 하자면, 파이썬에서는 hash 충돌을 막기 위해 key 값을 그대로 hash key로 구성하는 것이 아닌 내부적으로 indices라는 리스트를 이용하여 이 리스트 안의 값을 기반으로 새롭게 hash 값을 구성하여 해쉬 충돌을 막도록 구성했습니다.

11) [네카라쿠배] Stable sorting이란?

stable sorting이란, 정렬을 해야하는 리스트에 대해 중복 요소들에 대해서 순서가 바뀌지 않는 것을 보장하면 stable sorting, 보장하지 못하면 unstable sorting이라고 합니다. stable algorithm은 insertion, merge, bubble, counting sort가 이에 해당하며 unstable sorting은 selection, heap, shell, quick sorting이 있습니다.

12) [네카라쿠배] 파이썬에서의 해쉬 맵 내부구조를 설명하세요.

파이썬에서 dictionary가 해쉬 맵을 이용한 자료구조라고 볼 수 있습니다. 해쉬맵은 내부의 해쉬 함수를 이용하여 임의의 길이의 key 값을 고정된 길이의 데이터로 매핑하는 과정이 존재합니다. 파이썬 dictionary

는 해쉬 테이블에 이 고정된 길이의 해쉬 값, key, value 형태로 해쉬 테이블을 저장하고 있습니다. 하지만, 임의의 값을 해싱하여 해쉬 값을 만들면, 해쉬 값이 충돌이 되는 경우가 존재하는데 이럴 때에는 체이닝 기법을 통해 해결할 수 있습니다. 즉, 동일한 해쉬 값을 갖는 자료구조를 연결리스트 형태로 저장하여 충돌이 일어났을 때, 해당 노드의 next를 새롭게 연결해줌으로서 해결할 수 있습니다.

13) [네카라쿠배] 파이썬에서의 멀티 스레드의 단점

파이썬에는 다른 언어와 달리 global interpreter lock이라는 개념이 존재합니다. 한 자원에 대해서 한 스레드에게 할당한 후, 그 스레드가 끝날 때까지 lock을 걸어 다른 스레드가 접근하지 못하게 하는 것입니다. 때문에 파이썬에서 멀티 스레드 프로그램을 만들게 되면 동시에 일하는 것으로 바라지만 실질적으로 cpu를 점유할 수 있는 스레드는 한 개이기 때문에 실행속도가 오히려 단일 스레드보다 lock 연산이 생겨 더 느려집니다.

여기서 I/O 작업은 cpu 작업이 아니라, 이런 영향을 받지 않습니다. 그래서 큰 파일을 다운로드 받거나, 큰 입출력 등의 경우에서는 멀티 스레드를 이용하면 속도를 개선할 수 있습니다.

14) [스타트업, 네카라쿠배] 클래스는 무엇이며, 클래스의 장점은 무엇인가요?

클래스를 통해 여러 객체들을 일반화된 표현으로 규정함으로, 절차지향언어와 달리 소프트웨어 개발 속도 향상에 기여할 수 있습니다. 클래스를 붕어빵 틀이라고 하고 객체를 붕어빵이라고 한다면, 클래스는 변하지 않고 객체인 붕어빵에 팥을 넣거나, 슈크림을 넣는 듯이 이로써 생성된 객체도 전체적인 틀은 하나의 클래스지만 각 객체는 커스터마이징 할 수 있다는 점이 있습니다. 또한, 다른 이점으로는 부모 클래스 기능을 가진 자식 클래스가 여러개 있다고 하였을 때, 각 자식 클래스는 부모 클래스의 속성을 그대로 물려 받는 특징이 있으며, 이로 인해 기능 확장에 엄청난 이점이 있습니다. 또한, 추가적으로 부모의 특정 메서드를 재규정하여 자식 클래스에 맞게 조정하는 작업을 오버라이딩이라고 하며, 이 오버라이딩이 객체지향언어의 꽃이라고 생각합니다.

2. CS

1) [네카라쿠배] 자바에서의 string과 string buffer의 차이

2) [스타트업, 네카라쿠배] 자바에서의 상속에 대해서 설명

상속이란 객체지향언어에서 가장 핵심적인 개념 중 하나로써, 부모 클래스에서 정의한 기능을 그대로 자식 클래스가 물려받아서 사용한다는 개념입니다. 자식 클래스에서는 부모 클래스에 있던 모든 기능들을 그대로 가져와 추가 기능을 개발하거나, 오버라이딩을 통해 부모 클래스의 특정 메소드를 고도화 하는 등의 기능을 수행함으로 소프트웨어의 재사용성을 극대화 시키고, 개발 속도를 절차지향언어에 비해 비약적으로 향상 시킬 수 있습니다.

3) [스타트업, 네카라쿠배] 자바에서의 추상클래스는 인스턴스 생성이 가능한가?

불가능합니다. 추상클래스는 미완성된 추상메소드를 갖는 클래스로써 이를 상속받는 자식 클래스에서 실체화를 시키지 않으면 new A();를 했을 때 에러가 발생하게 됩니다. 그렇기 때문에 추상 클래스는 자식 클래스가 부모 클래스에서 특정 함수를 반드시 오버라이딩하여 구현하게 하고 싶을 때 사용하는 방법입니다. 사실 자바에서 인터페이스라고 하는 개념도 존재하는데 인터페이스도 마찬가지로 미완성된 메서드를 정의하고 인터페이스를 사용하는 클래스에서는 반드시 이를 실체화시켜야 사용할 수 있습니다. 소프트웨어공학 관점에서 보았을 때 추상클래스와 인터페이스에서 대부분의 경우에서 양방향으로 교환이 가능합니다. 하지만, 이런 공통점이 있음에도 자바에서 추상클래스와 인터페이스가 분리되어 있는 이유는 “일반화된 개념으로 묶을 수 있으면” 추상 클래스를 이용하는 것이고, “일반화된 개념으로 묶지 못하고 행위적인 일관성으로 묶을 수 있으면” 인터페이스를 이용하는 것이 바람직합니다. 예를 들어 사람, 개, 고래는 동물이라는 일반화 개념으로 묶을 수 있으므로 추상클래스를 이용하는 것이 바람직하고, 뇌, 컴퓨터, 계산기는 일반화된 개념으로는 묶지 못하지만 “계산을 할 수 있다.”라는 행위적 일관성이 있기 때문에 이런 경우에는 인터페이스로 구성하는 것이 바람직합니다. 한 가지 좀

더 말씀드리자면, 인터페이스의 대부분 네이밍의 마지막 부분의 able로 끝나는 것을 확인할 수 있는데, 이는 행위적 일관성을 표현한 것이라고 볼 수 있습니다.

4) [스타트업, 네카라쿠배] 자바에서의 오버라이딩과 오버로딩의 차이

오버로딩은 다형성을 지원하는 방법으로써 같은 이름의 메서드에 대해서 매개변수의 유형과 개수를 다르게 설정하여 런타임시 전달 받는 매개변수 유형에 따라 같은 이름의 메서드 목록에서 특정 함수로 넘겨주는 역할을 수행합니다. 즉, 메서드의 이름은 같으나 매개변수와 유형을 다르게 하여 다양한 유형의 호출에 응답할 수 있게 하는 테크닉이라고 알고 있습니다.

오버라이딩은 오버로딩과 전혀 관계없는 개념으로써 부모 클래스가 갖고 있는 메소드를 자식 클래스에서 재정의해서 사용하는 기술입니다. 그래서 부모 클래스의 해당 메소드는 자식 클래스의 메소드에 의해 덮어지게 되며 오버라이딩된 메서드에 경우에는 항상 자식 클래스에서 정의한 메소드가 호출이 됩니다. 여기서 항상 자식 클래스 메서드가 호출이 되므로, 부모 클래스의 메서드를 호출하고 싶은 경우에는 super라고 하는 키워드를 이용하면 부모 클래스의 메서드를 호출할 수 있습니다. 그렇기 때문에 특정 기능에서 부모 클래스에서는 해당 기능을 수행하기 위한 초기 세팅을 구현하고 오버라이딩한 자식 클래스에서는 가장 먼저 super 키워드로 초기화를 수행한 다음 이어서 작업하는 테크닉을 사용할 수도 있습니다.

5) [네카라쿠배] 자바에서의 부모 클래스로 업캐스팅한 자식 클래스의 레퍼런스의 오버라이딩 메소드가 호출되면 부모 클래스 메소드가 호출이 되나, 자식 클래스의 메소드가 호출이 되나?

당연히 자식.

6) [네카라쿠배] 자바에서의 접근 지정자와 각 특징을 설명

7) [네카라쿠배] 자바에서의 final과 static 설명

3. 데이터 전처리 및 AI 기초 지식

1) [공통] 데이터셋이 부족할 때에는 어떻게 해결할 수 있을까?

Augmentation을 통해 데이터를 더욱 확보하는 것이 가장 기본적인 전략이라 생각합니다. 저의 경우에 Augmentation를 할 때 총 2가지 전략으로 접근해보았습니다. 가장 먼저, 실시간 균열 탐지를 할 때 크롤링을 가장 먼저 해보았으나, 크롤링을 하고 나서 라벨링을 수기로 해주어야 한다는 단점이 존재했습니다. 그 다음의 전략으로 생성된 모델을 통해 아직 라벨링이 결정되지 않은 이미지에 대해서 자동으로 RoI를 찾아 라벨링하는 semi-supervised를 수행하여 자동화 도구를 만들었습니다. 하지만, 그 당시 대회 시작이 코 앞이였기 때문에 semi-supervised로 뽑아낸 라벨 데이터를 추가적으로 학습할 시간이 부족하여 적용해보지 못해 성능 향상이 이루어졌는지 확인이 불가능했습니다.

2) [공통] Optimazor는 무엇인가요?

옵티마이저는 경사하강법을 진행할 때 weight를 조정시켜주는 알고리즘입니다. 옵티마이저에는 크게 2종류의 방법이 있습니다. 가장 먼저 non-adaptive learning rate 방식으로 SGD, Momentum 등이 있습니다. 이 두 알고리즘의 특징은 학습시 고정된 learning rate를 사용한다는 것입니다. 여기서 SGD는 값을 계산할 때 전체 데이터셋으로 계산하는 것이 아닌, 일부 샘플들만 떼어와서 그 샘플 데이터셋으로 계산하는 방법입니다. 이런 특징으로 전체 데이터셋으로 계산하는 것보다 훨씬 더 빠른 수렴속도를 보여주며, 샘플들의 특성을 잘 파악해 GD 보다 학습이 잘되게 합니다. 하지만 SGD는 현재 기울기의 정도만을 보고 다음 스텝을 결정하므로, local minima의 기울기가 바뀌는 구간을 잘 빠져나오지 못합니다. 그래서 이를 해결해주기 위해 기존의 진행 방향에 대한 관성의 힘을 부여하자라는 모멘텀을 도입하여 이전 상태의 진행방향을 추가적으로 고려해줌으로써 좀 더

학습이 원활하게 수행되게 만들었습니다.

그 다음으로 adaptive learning rate 방식인 RMSProp, Adam이 있습니다. RMSProp은 학습이 진행되면 진행될수록 learning rate 값을 조정해나가며 좀 더 섬세하게 global minima를 찾아갈 수 있도록 합니다. 여기서 가장 널리 쓰이고 보편적인 아담은 RMSProp과 모멘텀을 결합한 기법입니다. 모멘텀처럼 관성의 법칙을 부여하고 RMSProp의 adaptive learning rate 방식을 결합한 옵티마이저라고 생각합니다.

3) [공통] train, test, validation set은 무엇이며, 왜 분할할까요?

train은 말 그대로 학습을 위한 데이터셋으로써, 모델의 학습에 사용되는 데이터셋이며

validation set은 학습된 모델의 성능을 검증하기 위해 사용됩니다. 그래서 train set의 일부를 validation set으로 뽑아냄으로써 모델의 overfitting을 확인하는 데에 사용됩니다.

마지막으로 test set은 모델의 일반화 성능을 평가하기 위해 라벨링이 없는 상태로 모델이 이를 예측함으로써 최종 모델의 성능을 검사하기 위해 사용되는 데이터셋입니다.

4) [공통] 모델의 overfitting이 일어났을 때 어떻게 해결할 수 있을까요?

overfitting이 일어났을 때 가장 보편적인 방법으로 드롭아웃 레이어를 추가하여 모델이 특정 feature들에 대해서만 과도하게 의존하지 않도록 학습시 일부 뉴런을 비활성화 상태로 만드는 것입니다. 그 다음으로는 feature의 특징에 비해 모델이 과도하게 깊하게 구성이 되어 있는 경우에도 과적합이 발생합니다. 그렇기 때문에 모델을 좀 더 단순하게 표현해보거나, feature가 너무 많아 과적합이 발생할 수 도 있으므로, feature engineering을 통한 feature를 조정해주는 것도 방법이 될 수 있습니다. 그 외로는 L1, L2 정규화 등을 적용하여 특정 weight들에만 가중치를 높게 주는 문제를 해결하여 과적합을 방지할 수 있습니다. 다음으로 early stopping과 batch normalization를 수행하여 gradient smoothing 효과를 줌으로 오버피팅에 어느정도 대처할 수 있습니다.

5) [공통] Batch Norm에 대해서 설명해주세요.

배치 놈은 인풋 노말리제이션에서 수행했음에도 불구하고 레이어를 통과하면 통과할수록 input 디스트리뷰션이 어그러지는 interval coverate shift 문제를 해결하기 위해 나온 기법입니다. 이 기법은 각 레이어에서 인풋 노말리제이션을 수행한 것처럼 정규화를 레이어마다 특정 trainable parameter인 감마와 베타를 두어 데이터 특성에 맞게 다시 수행해주는 기법입니다. 그렇기 때문에 배치 놈은 regularization의 효과를 발휘할 수 있으며 overfitting에도 어느 정도 효과가 있는 것으로 알려져 있습니다. 또한, 데이터의 분포를 계속해서 유지해줌으로써 gradient smoothing 효과로 비교적 모델이 global minima에 더욱 빠르게 수렴할 수 있도록 도움을 줍니다. 하지만, feature에 계속된 변형을 가하므로, 원래 데이터가 가지고 있던 특성이 사라진다는 특징이 있습니다. 그렇기 때문에 대표적으로 GAN의 생성자와 판별자 부분에는 배치 놈을 사용할 수 없도록 권장하고 있는 것으로 알고 있습니다.

4. 머신러닝

1) [스타트업] 크로스 엔트로피 수식에 대해서 설명해주세요. Information gain은 어떤 식으로 하는 건가요?

2) [공통] 클래스 불균형 문제를 풀기 위해서 프로젝트에서 Focal loss를 사용하셨던데, 수식은 어떻게 되며, 어떤 원리로 Focal loss를 쓰셨는지

3) [스타트업, 네카라쿠배] 랜덤 포레스트에 대해서 알고 계신만큼 다 설명해주세요.

4) [스타트업] 다중공선성은 무엇이며 이 부분에 대해 왜 고려해야 하는지와 어떻게 해결하는지 구체적으로 설명해주세요.

다중공선성이란 여러 개의 벡터가 선형적으로 같은 방향을 가리키고 있는 문제입니다. 머신러닝의 대부

분의 기법은 선형성과 독립성을 가정하여 계산이 이루어지도록 되어 있습니다. 여기서 x인 여러 개의 독립 변수가 다중공선성이라 같은 방향을 가리키고 있다고 하면 이런 독립변수라고 하는 기본적인 가정이 어긋나 버리기 때문에 다중공선성 문제를 꼭 해결해주어야 합니다. 이를 해결해주는 방법으로써는 상관관계를 뽑아내어 상관관계가 높은 feature간의 조합을 삭제해줌으로써 다중공선성 문제를 해결할 수 있습니다. 경험적인 측면에서 상관관계가 0.7 이상인 feature들은 다중공선성의 문제가 있다고 판단하여 제거해주는 편입니다.

5) [대기업] 허프 변환에 대해서 설명해주세요.

허프 변환은 전통적인 컴퓨터비전에서 사용하는 기법으로써 주로 직선이나 원을 검출할 때 사용하는 알고리즘으로 알고 있습니다. 이는 하나의 점을 허프 공간으로 매핑을 하면 한 직선으로 표현될 수 있다라는 가정을 가지고 있습니다. 그래서 여러 점에 대한 직선을 구하고 이 직선의 교점이 되는 부분을 찾아서 여러 점과 병합을 수행함으로써 직선이나 원을 검출하는 알고리즘입니다.

6) [대기업, 네카라쿠배] HOG Detection에 대해서 설명해주세요.

HOG는 딥러닝을 이용한 object detection를 사용하기 이전에 가장 활발하게 사용되었던 객체 검출 알고리즘 중 하나입니다. 이미지에 대해 지역적인 filter를 사용하여 영상의 특징을 뽑아내고, 이에 대한 히스토그램을 기반으로써 pixel-by-pixel로 기울기인 gradient를 구하게 됩니다. 여기서 인접한 gradient 병합함으로 object feature를 구하는 방식의 알고리즘으로 구현되어 있는 걸로 알고 있으며, 주로 object tracking에서 벤치마킹하여 사용하는 기법으로 알고 있습니다.

5. 딥러닝

1) [공통] CNN에 대해서 아는만큼 이야기 해주세요.

CNN은 Convolution filter를 기반으로 이미지 같은 고차원의 shape 변경 없이 사용함으로써 이미지 분야에 대한 퍼포먼스가 막강한 기법입니다. filter가 local하게 연산이 이루어지므로, local feature를 추출하는 데에 다른 기법보다 훨씬 효율적이며, 기본 inductive-bias 속성으로는 locality와 translation equivariance가 존재합니다. 여기서 말씀드린 locality는 convolution 연산이 지역적으로 수행이 되는 점이며, translation equivariance의 경우, 객체의 위치와 관계없이 filter의 parameter가 공유되어 위치 불변의 특징을 추출한다는 점이 CNN의 가장 큰 특징이라고 할 수 있습니다.

2) [공통] RNN에 대해서 아는 만큼 설명해주세요.

RNN은 각 히든 레이어의 컨텍스트 벡터에서 출력 값을 기반으로 아웃풋을 만들어내며, 이 아웃풋을 다음 레이어의 입력 값의 일부분으로 사용한다는 특징이 있습니다. 이런 특징을 기반으로 시계열 데이터를 다루는 주제에서 가장 적합한 기법이지만, 각 컨텍스트 벡터가 순차적으로 연결이 되어 있기 때문에 특정 히든 레이어 기준으로 최근 데이터가 더 강하게 학습되고 레이어가 깊어지면 깊어질수록 과거 데이터가 희미해진다는 단점이 존재합니다. 그래서 RNN은 시계열에서 직전 상태만을 주로 고려하는 경우에서 사용할 수 있는 알고리즘입니다.

그 다음으로 LSTM의 경우, gate의 개념을 부여하여 특정 게이트는 최근 정보를 학습하고 다른 게이트는 과거 데이터를 위주로 학습하게 하여 기존의 RNN에서 발생했던 문제점을 해결한 기법입니다. 여기서 forget gate의 경우, 특정 주기마다 weight 들을 초기화를 해줌으로써 긴 주기의 추론에서 특정 weight만 과도하게 업데이트 되는 것을 방지하는 regularization 역할을 해주며, gradient vanishing이나 exploding 문제를 해결해주는데 큰 기여를 해줍니다.

3) [공통] Transformer에 대해 아는 만큼 설명해주세요.

대부분의 RNN의 경우에는 히든 레이어의 컨텍스트 벡터가 순차적으로 연결되어 있는 구조로써, feature들이 유의미하게 순차적으로 연결된 경우에서만 효과적인 성능을 보인 반면에, transformer의 경우 self-attension의 개념이 부여되어 이런 컨텍스트 벡터가 대칭적으로 연결이 되어 있는 점이 가장 큰 특징입니다. Transformer은 self-attension으로써 각 encoder의 컨텍스트 벡터를 각각 연결해주거나, 각 encoder와

decoder 간의 모든 컨텍스트 벡터를 연결해주는 등의 기법을 사용할 수 있습니다. 이런 self-attension 때문에 입력 시퀀스가 유의미하게 순차적으로 연결이 되어 있지 않아도, 효과적으로 데이터를 처리할 수 있다는 장점이 존재합니다. 하지만, 모든 입력 시퀀스를 임베딩하여 변환함으로 inductive-bias가 낮아져 대부분의 경우 많은 양의 학습 데이터를 필요로 하는 단점이 있습니다.

6. 기타 질의

1) [스타트업] 저희 회사에서 저장된 데이터들의 떠오르시는 DB 모습을 다 이야기 해주시고 왜 그렇게 되어 있을 것 같으신지 말씀해주세요.

2) [스타트업] 저희 분야에서 데이터를 분석하는 데에 있어 내부 데이터 측면에서 어떤 문제점을 내포하고 있을 것 같나요?

3) [스타트업] 플랫폼 비즈니스와 생산 중심의 비즈니스의 차이는 무엇이라고 생각하시나요?

4) [스타트업] 다음은 저희 회사가 풀어야 할 문제입니다. 해당 백그라운드를 들으시고 2분 동안 생각할 시간을 드리겠습니다. 저희 회사 특성상 수요와 공급이 불일치한다는 점이 존재하는데, 이런 문제를 규정해주시고 어떻게 하면 저희 회사가 이런 문제를 해결할 수 있을까요? [대부분의 스타트업 기업은 이런 창의적인 질문임.]

5) [스타트업, 네카라쿠배] 부동산 예측을 진행하는 데에 있어서 변수가 어떤 식으로 구성이 되어 있는지 이야기 해주시면서 저 프로젝트가 실 서비스 가치가 있다고 생각하시는 지 의견이 궁금합니다.

6) [스타트업, 네카라쿠배] 그렇다면, 저 서비스를 다시 한 번 해볼 수 있는 기회가 주어졌을 때 어떤 식으로 고도화를 할 수 있을 것이라 생각하시나요?

7) [스타트업, 네카라쿠배] 저 프로젝트가 현재 시점의 적정 가격을 알려주는 것인지, 미래의 시점에서 적정 가격을 예측해주는 것인지 궁금합니다.

8) [스타트업, 네카라쿠배] 면접관이 판단했을 때 당연히 면적이나 위치가 가장 중요한 피처로 설정이 됐을 것 같은데, 모델의 해결 결과는 어떻게 됐나요?

9) [공통] 같이 일하고 싶은 사람은 어떤 사람인가요?

10) [대기업] 2분 자기소개 해주시기 바랍니다. 지원동기를 포함해서 이야기 해주세요.

11) [대기업] 오토인코더 기반 악성코드 탐지 연구를 하셨었는데, 이 연구에 대해 데이터셋 취득, 처리, feature 설정, 사용한 모델의 구조와 특징들을 다 상세하게 이야기 해주세요. [연구를 진짜로 진행했는지 확인하기 위함.]

12) [대기업] 삼성서울병원에서 했던 시계열 데이터 분석 연구에 대해서 구체적으로 설명해주세요. [연구를 진짜로 진행했는지 확인하기 위함.]

13) [대기업] 그럼 병원에서 했던 연구의 결과를 보고 주변인들이 뭐라고 반응하던가요? [연구에 대한 성과를 확인하기 위함.]

14) [대기업] 이력서를 보니 연구 경험이 풍부하신 것 같은데, 어떻게 해서 이런 연구들을 하게 되었으며 각 연구에 대해서 간단하게 설명해주세요.

15) [대기업] 우리 회사에 대해서 아는 만큼 이야기 해주고, 만약에 입사한다면 무슨 일을 하게 될지에 대해서 말씀해주세요.

16) [대기업] 본인이 생각하는 DevOps는 무엇인가요? 그렇다면 MLOps는요?

: 데브옵스는 말그대로 개발 환경을 관리하는 업무로써, 컨테이너 오케스트라 등의 툴을 사용함으로써 개발 환경과 배포 환경의 자동화, 효율성 재고 등에 기여할 수 있으며, 좁은 의미의 데브옵스는 크게 서버, 클라우드 등의 각 자원들을 의미합니다. 그래서 데브옵스를 수행하면서 가장 중요한 점은 각 자원들에 대한 트러블슈팅과 환경 세팅이 가장 중요하며, 어떻게 하면 가장 효율적인 배포 환경을 만들 수 있을지가 가장 중요한 점이라 생각합니다. MLOps는 데브옵스에서 한층 더 진화하여 ML/DL 모델을 탑재한 환경을 의미합니다. 이 경우에는 GPU와 모델의 Inference 서비스 등에 대한 고려가 더욱 이루어져야 하며, kafka, hadoop 등의 프레임워크를 적극 채용하며 가장 효율적으로 데이터 파이프라인과 인퍼런스 파이프라인을 구축하는 것이 가장 중요하다고 생각합니다.

17) [대기업] 자신이 생각하기에 성공적인 회사 생활이 무엇인지에 대해 규정하고, 이를 이루기 위해서는 무엇을 해야할지 이야기 해주세요.

: 회사의 공통의 목표를 이루기 위해 협업함으로써 신뢰성 있는 서비스를 고객사에게 제공하는 것입니다. 다만 이 이야기가 굉장히 추상적으로 들리실 수도 있는데, 고객에게 신뢰성 있는 서비스를 제공하는 것은 이는 잠재 고객을 충성 고객으로 만들 수 있다는 의미이며, 이 고객과 새로운 비즈니스 관계가 연결이 됩니다. 이는 최종적으로 회사의 입장에서는 매출 증대로 이루어질 수 있으며, 엔지니어 입장에서는 본인의 가치를 더욱 더 끌어올려줄 수 있습니다. 이런 워너비를 이루기 위해서는 팀원 과의 협업 관계를 계속해서 유지해야 하며, 효율적인 개발을 위해 끊임없이 자기 계발에 투자해야 한다고 생각합니다. [회사 인재상에 입각하여 스토리텔링]

18) [대기업] 자신이 가장 자신 있는 프로그래밍 언어를 선택하고, 그 언어에 대한 장단점을 이야기 해주세요.

: 파이썬. 파이썬은 커뮤니티가 가장 활발하여 다양한 모듈들과 기술 지원을 받을 수 있다는 장점이 있으며, 그 외에도 파이써닉한 스타일로써 리스트 컴프리헨션, 제너레이터 등을 통한 기법을 통해 다른 언어보다 function call를 비약적으로 낮추면서 메모리를 효율적으로 쓸 수 있다는 장점이 존재합니다. 하지만, 인터프리터 언어의 최고 단점인 global interpreter lock이 존재하여, 다중 스레드 환경에서 효율성이 크게 좋지 못하다는 단점이 있습니다.

19) [공통] 우리 회사에 대해서 얼만큼 알고 있는지 상세하게 설명 부탁드립니다.

20) [대기업] 코로나 진단 키트 관련 AI 모델을 만든다고 하였을 때, 가장 집중해야할 성능 지표(메트릭)이 무엇인지 설명하고 선택한 이유를 납득이 될 수 있게 설명해주세요.

sensitivity - 코로나 진단 키트의 경우, 실제 양성을 양성으로 올바르게 분류하는 것이 가장 중요하니 민감도가 가장 중요할 수 밖에 없음. 음성을 양성으로 오분류하는 경우에는 병원 등에서 추가 검사를 받으면 되므로, specificity의 경우에는 어느 정도의 오탐율을 허용해도 된다고 생각함.

21) [대기업] 파이썬의 yield에 대해서 아는만큼 상세하게 설명해주세요. (제너레이터 관련)

yield를 사용하면 일반 변수가 나오는 것이 아닌 제너레이터 객체가 반환이 됩니다. 제너레이터는 미리 객체를 만들지 않고 실시간으로 계산하기 때문에 메모리를 훨씬 더 효율적으로 사용한다는 특징이 있습니다. 예를 들어, 일반 반복문으로 1 ~ 10000000까지 돌며 변수를 만드는 경우, 이미 전체 숫자가 메모리에 올라가 있는 반면, 제너레이터를 사용하는 경우 실시간으로 계산하여 변수를 생성하여 저장하므로, 메모리를 훨씬 더 아껴

쓸 수 있는 장점이 존재합니다.

22) [대기업] C++의 스마트 포인터는 무엇인지 설명해주세요.

메모리 누수를 방지하기 위해, 사용이 끝난 포인터의 메모리를 자동으로 해제해주는 포인터입니다.

23) [공통] 머신러닝의 베이지안 확률에 대해서 설명해주세요.

24) [대기업] 파이썬에서는 메모리 누수 현상이 발생할까요? 예 아니오로 대답하고, 그 이유를 설명해주세요.

발생할 것 같습니다. 기본적으로 하나의 프로세스에는 여러 영역으로 구성이 되어 있는데, 여기서 프로그램 스택 영역에 공격자가 악의적으로 특정 함수를 계속해서 호출하는 경우 프로그램 스택이 다 차게될 것이고 추후에는 가능한 영역을 넘어 버리는 누수 현상이 발생할 것으로 생각합니다. 아마 스택 오버 플로우 공격은 모든 언어에 다 가능한 기법으로 알고 있습니다. 이 부분에 대해 좀 더 공부해보겠습니다.

25) [대기업] 최근에 가장 흥미있게 읽은 IT 관련 서적이나, 뉴스에 대해서 설명해주세요.

우리 모두가 알고 있는 discord의 백엔드가 go에서 러스트로 바뀌었다는 글을 보았습니다. go의 경우, GC overhead라고 하는 현상이 생겨 메모리가 가득차게 되었을 때 자바의 가비지 컬렉션과 비슷한 느낌으로 잠깐 메모리 정리를 하는 현상이 발생하게 됩니다. 하지만 discord는 실시간으로 폭발적인 트래픽이 쌓이는 프로그램으로써 이런 오버헤드를 더 이상 감당할 수 없어, 완전히 저수준의 언어인 러스트를 채용함에 따라 이런 문제를 해결할 수 있다고 보았습니다. 이 글을 보며, 엄청난 트래픽이 오가는 경우에 대해 프로그램을 작성할 때 개발 언어를 선택하는 것 마저 엄청나게 중요한 역량이라는 것을 느끼게 되었습니다.

26) [대기업] 개발을 하는 데에 있어서 라이브러리나 외부 모듈들을 선택을 할 때, 어떤 기준으로 이런 것들을 선택하시는지 알려주세요.

저는 가장 먼저 프로그램 요구사항에 맞는 라이브러리를 선택합니다. 특정 주제에서는 텐서플로우가 더 효율적인 경우가 있고 다른 주제에서는 파이토치가 더 효율적인 경우가 있습니다. 이런 요구사항을 따지며 프레임워크들을 선택하며, 모듈의 경우에는 다른 사람들이 가장 많이 사용하는 well-known 모듈을 사용합니다. 그 이유는 많은 사람들이 사용하기 때문에 안정성이 가장 높다고 생각했으며, 이 모듈을 사용할 때도 가장 reliable한 버전을 사용합니다. 예를 들어, opencv 1.5버전이 최신 버전일 때, 이 최신 버전은 아직 잠재적인 위험이 무엇이 있는지 알 수가 없어, opencv 1.3 버전이 가장 reliable한 버전이라 했을 때 해당 버전을 사용해서 개발합니다.

27) [공통] 프로젝트를 진행하는데 가장 어려운 부분이나 가장 고려해야할 부분은 무엇인가?

데이터 분석을 수행할 때 가장 중요한 것은 데이터의 품질이라고 생각합니다. 이 데이터의 품질은 데이터에 결측치가 많다거나 이상치가 많은 것을 의미하는게 아니라 데이터가 정형화된 형태로 존재하는가?를 이야기하는 것입니다. 하지만, 이 프로젝트의 환자의 데이터는 모두 양식과 포맷이 달랐으며 이런 문제를 해결하기 위해 데이터 전처리를 하기 전에 데이터 포맷을 모두 동일하게 맞춰 주는 SW를 개발했고 이런 문제를 해결하기 위해 알고리즘 구상까지 하는 것이 가장 어려웠었습니다.

28) [공통, 대학원] 삼성서울병원을 그만 두게 된 계기나 목적이 있었는지?

사실 저는 삼성서울병원에서 연구를 진행하며 소위 말하는 [연구를 위한 연구]를 하며 이 환경에 익숙해지는 제 모습이 싫었습니다. 이런 연구를 진행하며 서비스 가능한 AI 솔루션 개발에 대한 공부를 더욱 하고 싶어, 삼성서울병원에서 더욱 나은 처우로 근무를 계속하자는 오퍼가 들어왔음에도 거절하고 지금은 이런 공부를 계속해서 하고 있습니다.

29) [공통, 대학원] 삼성서울병원에서 DevOps도 했는데, 구체적으로 무엇을 했는지 또한 역할 분담은 왜 이렇

게 된 것인지?

사실 삼성서울병원은 연구만을 위한 환경으로 구성되어 있었습니다. 그래서 어느 누구도 자기가 사용하는 분석 환경을 관리하지 않으려고 했습니다. 이런 과정이 지속되며, 저희 팀 팀장님이 자기 팀에서 관리하겠다 하며 혼자서 관리를 하시다가, 저를 면접을 보시고 DS적인 지식과 CS 지식이 충분히 갖춰져 있음을 판단하시고 저와 같이 원내 온-프레미스 서버와 성균관 대학교 협업 연구망을 관리하였습니다.

30) [공통, 대학원] 네이버 AI 부스트캠프는 어떤 식으로 운영하는 건가? 캠퍼들과 어떤 식으로 멘토링을 진행하는지, 그럼 거기서 제일 재밌는 활동은 무엇인지?

네이버 AI 부스트캠프에서 캠퍼 10명 내외 정도를 맡아 멘토링을 진행합니다. 좀 더 구체적으로 말씀드리면, 저는 선생님으로써 캠퍼들은 학생으로써 역할을 하며 캠퍼들 대부분이 현업에서 활동해본 경험이 없음을 인지하고 현업에서의 데이터 분석 방법과 플로우, 트랜드는 무엇이 있는지 팁들을 공유하고 캠퍼들의 학습을 하는데에 애로사항과 고민거리를 들어주고 대회를 진행할 때 캠퍼들이 만든 가설들을 점검해주고 피드백 해주는 활동이 가장 의미있고 재밌는 시간이라 생각합니다.

31) [스타트업] 데이터 사이언티스트로서의 내가 생각하는 최종 인재상은 무엇이라 생각하는가?

사실 이 문제는 데이터 사이언티스트 뿐만 아니라, IT 전체에 필수가 되는 지식이라 생각하는데요. 요즘 시대가 단순히 백엔드만을 잘 하는 개발자가 아닌 스스로 DevOps도 구축할 줄 아는 다방면한 인재를 원합니다. 이렇듯 데이터 사이언티스트도 단순히 데이터 분석만을 잘하는 것이 아닌 (데이터 분석을 잘 하는 사람은 널리고 널렸으니) 스스로 본인의 환경을 구축할 수 있는 MLOps 지식을 겸비한 데이터 사이언티스트가 되어야 한다고 생각합니다.

32) [스타트업, 네카라쿠배] 데이터 분석에 관련된 흥미가 있나요?

사실 AI 모델을 개발한게 아닌, 데이터 분석을 통해 인사이트를 도출했던 경험이 있습니다. 이력서에 보시면 [특정 지역 매매 대상 아파트 적정 가격 예측]이라는 프로젝트를 통해 실제 아파트 정보를 이용하여 이를 위치 등으로 시각화를 하면서 매매 가격이 여러 군집을 이룸으로 형성되는 것을 확인하며 데이터 분석이 이렇게 재미있는 것이였구나 라는 경험을 얻게된 계기였습니다.

33) [공통] 우리 회사에 대해 어떻게 관심을 가지게 되었는지?

34) [공통] 갈등이 발생했을때 어떻게 대처 했는가?

~~한 경험이 있습니다. 이를 ~~해서 해결 했습니다. 이렇듯 회사에서 근무하며 갈등이 발생하는 경우가 많을 것입니다. 그런 상황에서 저의 특유의 갈등 해결 능력을 발휘하여 본 회사에서 커뮤니케이션의 중심이 되는 사람이 될 수 있도록 노력하겠습니다.

35) [공통] 연구를 수행하는데에 있어서 본인만의 특징 또는 장점

연구를 진행하는 데에 있어 추진력이 가장 큰 강점이라 생각합니다. 저는 EDA를 통해 데이터의 특징을 발견하고 이런 특징에 따른 어떤 모델을 선택해야 하는지에 대해서 깊게 생각하며 특히 연구를 진행하기 앞서 선행 논문들을 탐색하며 이 기법을 우리 프로젝트에 적용할 수 있는지, 고도화에 사용될 수 있는지 가능성을 알아보기 위해 프로젝트에 항상 적용을 시켜보며 성능을 검증하는 특징이 있습니다.

특정 회사 - 대학원 계약학과 면접

1. 1분 자기소개 및 지원 동기
[첫 시작은 최대한 컴펙트하고 자기의 전문 분야로 어필하는 것이 좋음]
e.g. 저는 ##와 @@에 관심이 있습니다. ##은 !! 연구 주제에서 ~~한 성과를 달성하며 경험을 쌓았으며, @@은 ** 관련 대회에서 최우수상을 받으면서 실력을 쌓아 나갔습니다. 이런 연구 경험은 ~~가 %%한 사업 분야에 ~~하고 있어, 이런 부분에 대해 저의 연구 역량을 가장 잘 발휘할 수 있는 회사라 생각되어 지원했습니다.

[두 번째 문단은 회사에 대한 관심을 어필하고 인재상에 맞게 이야기]
~~은 ~~ 부문에서 ~~를 적극 도입하여 ~~한 혁신을 달성하고 있는 혁신적인 회사입니다. ~~에 대해 AI를 적극 도입하며 고객에게 좀 더 윤택한 삶을 제공하고, 더 나아가 스마트 ~~ 분야에 혁신을 계속해서 시도하는 모습에 영향을 받았습니다. 저는 ~~~ 한 사람으로 ~~에서 ~~한 사람이 될 수 있도록 끊임 없이 ~~ 하겠습니다.

[마지막은 포부]
여태까지의 연구 경험을 토대로 ~~에서 ~~한 연구를 수행 or ~~한 플랫폼을 개발하며 제가 가장 좋아하는 ##와 @@ 분야의 지식을 다져나가며 ~~(회사의 인재상)한 사람이 되겠습니다.

2. 전공 과목
2-1: 전공 과목 중 가장 재미있었던 과목과 그 이유, 혹은 가장 싫어했던 과목과 그 이유

2-2: ##대학원 인공지능학과에서 연구를 할 때의 본인의 강점은 무엇인가?
: 목표지향적인 성격으로써 연구를 가장 효율적으로, 완벽하게 수행하기 위해 여태까지 연구에서 관련 선행 논문들을 먼저 탐색하여 기초 지식을 탄탄히 하고 내 연구에서는 어떤 식으로 접근할 것인지에 대한 사전 조사를 완벽하게 수행합니다. 또한, 창의적인 성격으로 연구에서 다른 사람들이 미처 하지 못했던 새로운 가설을 토대로 혁신적인 연구를 수행한 경험이 있습니다. 이 연구는 ~~~ 했으며, 이 외에도 여러 연구를 통해 논문과 학술대회에서 구두 발표를 수행한 경험이 있습니다.

제가 언급한 경험들은 대학원에서 연구를 하며 가장 중요한 역량이라고 생각되며, 그 외에도 ~~한 역량이 있기 때문에 AI 관련된 분야 뿐만 아닌 ~~에서 적절히 기여할 수 있다고 생각합니다.

2-3: 해당 연구실을 선택한 이유?

2-4 : 그러면 대학원에서 하고 싶은 연구는 무엇인가요?

2-5: 이 연구실에서 연구한 주제를 우리 회사의 어떤 부분에 어떻게 활용할 수 있다고 생각하시나요?

2-7 졸업 후에 우리 회사에 입사한다면 무슨 일을 하고 싶은가요?