**OCR 인식 및 활용 방안**

* **기능 정리**

1. Optical Character Recognition(OCR)
   * **TEXT\_DETECTION**: response includes the detected phrase, its bounding box, and individual words and their bounding boxes
   * **DOCUMENT\_TEXT\_DETECTION**: response includes additional layout information, such as page, block, paragraph, word, and break information, along with confidence scores for each. (The sample below uses a simplified example; the response from a dense document is too long to display on this page.)
2. **LABEL\_DETECTION**: detect and extract information about entities within an image, across a broad group of categories. Labels can identify objects, locations, activities, animal species, products, and more. Response includes the detected labels, their score, and an opaque label ID
3. **OBJECT\_LOCALIZATION**: detect and extract multiple objects in an image with Object Localization. Identifiy multiple objects in an image and provides a LocalizedObjectAnnotation for each object in the image. Each LocalizedObjectAnnotation identifies information about the object, the position of the object, and rectangular bounds for the region of the image that contains the object.
4. **CROP\_HINTS**: return the coordinates of a bounding box that surrounds the dominant object or face in an image. These coordinates can be used to crop the image to feature that dominant object.
5. **FACE\_DETECTION**: detects multiple faces within an image and provides coordinates of key facial features. It also returns emotional state predictions (such as joy, anger, and surprise) and whether headwear is being worn.
6. **IMAGE\_PROPERTIES**: request returns the dominant colors in the image as RGB values and percent of the total pixel count.
7. **LANDMARK\_DETECTION**: detect well-known natural and human-made landmarks, and return identifying information such as an entity ID (that may be available in the Google Knowledge Graph), the landmark's name and location, and the bounding box that surrounds the landmark in the image.

* **사전 분석**

1. X, Y 값
   1. Type.**TEXT\_DETECTION**으로 지정한 경우의 분석
      * A TEXT\_DETECTION response includes the detected **phrase**, its **bounding box**, and **individual words and their bounding boxes**.
   2. Origin
      * Upper-left corner가 (0,0)의 좌표를 갖음
   3. 문서 DPI, font 크기에 따른 변화
      * 600x600 해상도와 300x300 해상도 파일에서 읽어오는 좌표는 다름
      * 예를 들어, (2157,777) 좌표가 (1080,387) 좌표로 인식됨 (🡺 해상도에 비례)
      * 결론: v1 단계에서는 표준화된 해상도로 image를 스캔!!
   4. Multi-page의 경우는?
      * Y의 값이 계속 증가, 가령 300x300 jpg의 경우 2 page에 있으면 Y값이 3108에서 시작

🡺 page를 인식하는 방법? 또는 굳이 page를 인식해야 할까?

1. 4개의 vertices 의미

대한민국

1st vertex (x1, y1)

4th vertex (x1, y2)

2nd vertex (x2, y1)

3rd vertex (x2, y2)

1. 다른 Type은 어떤 의미 및 결과를 주는지 테스트?
   1. Type.**DOCUMENT\_TEXT\_DETECTION**으로 지정한 경우의 분석
      * A DOCUMENT\_TEXT\_DETECTION response includes additional layout information, such as **page**, **block**, **paragraph, word**, and **break information**, along with **confidence scores** for each. (The sample below uses a simplified example; the response from a dense document is too long to display on this page.)
      * Document text detection can also **detect handwriting** in an image. When you know images contain handwriting, you can use a language hint to improve results.

* **준비/Layout 디자인**

1. 원본에 대한 Text 분석
   * GoogleObjects (GOs): objects in the format of Google JSON
   * RecognizedCpatureObjects (RCOs): 필요한 Google Object만 선별한 captured objects s
2. Page의 기준이 되는 정보 인식 (예를 들면, 양식의 title등, 다만 항상 정확히 인식할 수 있는 정보이어야 함)
3. Key(고정) field – 페이지, 위치, 크기 (이를 Block으로 지정), 고정 텍스트(이는 위치가 가변일 경우 활용)
4. Value field – 연관 key 정보, value 페이지, 위치, 크기 (이를 Block으로 지정), 속성(영문, 한글, 숫자, 특수문자, 기타 등)
5. Table 에 대한 처리 방안? 각 cell을 정의해야 하면 불편, 페이지 넘어가는 테이블?
6. X, Y의 위치, line별 Y값 증가를 인식하는 방법?

* **절차**

1. Page 기준이 되는 정보 검색하여 기준 위치와의 오차 파악 (보정을 위하여 필요)
   * 이 정보가 단일 word 또는 phrase일 수가 있으므로 대책 필요)
2. 전체 대상 object의 위치에 대하여 보정 작업 수행
3. Title text 및 위치 인식, 상응하는 value에 대한 위치(upper left, lower right)
4. 원본 지정된 해상도로 스캔하여 저장
5. 원본에 대한 Text 인식
6. Format이 고정일 경우 (scan되는 위치가 다를 수 있으므로 비현실적? 또는 보정작업으로 가능?)
   * 일정 영역(page, (x1,y1)~(x2,y2))을 지정 또한 속성 (예를 들면, 영문/숫자/영한/영한숫자 등)
   * Value field 범위에 있는 텍스트 찾기 및 순서대로 조합
   * 속성 비교하여 맞으면 OK
   * 속성이 틀리면 인식 오류일 경우 사용자 검토를 제시
7. Format이 가변일 경우
   * Key가 될 수 있는 title 부분을 지정하여 layout에 저장 (page, (x1,y1)~(x2,y2))
   * Value가 되는 일정 영역(page, 시작 상대 좌표 및 크기)을 지정 또한 속성 (예를 들면, 영문/숫자/영한/영한숫자 등)
   * key field에 해당하는 텍스트 찾아서 좌표(x,y) 인식
   * value field에 대한 절대 좌표 계산 (key field 좌표, value field 상대좌표 및 크기)
   * Value field 범위에 있는 텍스트 찾기 및 순서대로 조합
   * 속성 비교하여 맞으면 OK
   * 속성이 틀리면 인식 오류일 경우 사용자 검토를 제시

* **동일 문서에 대한 Image format별 비교**
* 대상 문서: 사업자등록증

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Format | 해상도 | Size(KB) | Google OCR 인식 후 | 비고 |
| JPG | 300 | 1,338 | 많은 부분에서 인식 오류 발생 |  |
|  | 600 | 2,244 |  |  |
| TIF | 300 | 98 | - | Format 오류 |
|  | 600 | 315 | 꽤 깜끔하게 인식됨 |  |
| PDF | 300 | 1,344 |  | 다른 방법으로 인식 |
|  | 600 | 2,253 |  |  |

* **Layout format**

“Doc-ID”: *<문서 ID>,*

[

{

“Seq”: *<순서>*,

“Key”: *<value>*,

Key value가 여러 vertex의 조합으로 이루어 진 경우 판독시 key value를 찾지 못할 것임.

그럼, 여기도 값과 영역을 가져야 하나?

“Target”: {

“x-relpos”: x 위치 증분,

“y-relpos”: y 위치 증분,

“width”: 대상 영역의 폭,

“height”: 대상 영역의 높이

}

},

{

..

}

]

* **Layout 정보 수집**

1. Scan 후 object (seq, value, x1, y1, x3, y3)를 수집
2. key 영역 (x1, y1, x3, y3)을 잡으면 해당 영역 내 존재하는 object를 선별하여 순서대로 key 값을 구성 (space 처리는?)
3. key 영역 (x1, y1, x3, y3)에 대한 target 영역을 잡으면 대상 key 영역에 상대적인 위치 및 크기를 정의
4. 문제: (해상도에 따른) Google Vision API에서 인식한 위치와 Layout designer에서 정의한 영역의 위치 mapping?
5. 위의 정보를 DB에 저장

* **Layout 정보 수정**

(Layout 수정 시 많은 논리적 사고 및 고려 사항 있음)

* **문서 인식**

1. DB에서 해당 문서의 layout 정보를 가져옴
2. Scan 후 object (seq, value, x1, y1, x3, y3)를 수집
3. Scan된 object들에서 key value에 해당하는 object를 검색 (조합 vs. 단독)
4. 대상 위치를 계산 (key value 위치(x,y)에 x, y offset 계산 및 폭, 높이 계산)
5. Scan object들에서 해당 영역에 해당하는 object를 검색하여 순서대로 정렬