
بسم الله الرحمن الرحیم

بازیابی اطلاعات

پروژه: ارزیابی مدل‌های فضای برداری و مدل‌های احتمالی زبانی

آرمان آزادی

زمستان ۱۴۰۴

فهرست مطالب

چکیده

این گزارش به پیاده‌سازی، تنظیم و ارزیابی یک سیستم بازیابی اطلاعات می‌پردازد که برای بازیابی اسناد مرتبط با پرسش‌های کاربر طراحی شده است. این سیستم تحت محدودیت‌های خاصی ساخته شده است، از جمله محدودیت اندازه واژگان به حداقل ۵۰۰ کلمه و ممنوعیت استفاده از کتابخانه‌های خارجی. سه مدل بازیابی متمایز پیاده‌سازی شدند: مدل فضای برداری(BM25) ، مدل زبانی تک‌کلمه‌ای (Unigram) با هموارسازی Dirichlet) (Smoothing، و مدل زبانی دوکلمه‌ای (Bigram) با درونیابی(Interpolation) . مدل‌ها بر روی یک مجموعه داده اعتبارسنجی (Validation Set) برای بهینه‌سازی ابرپارامترها (k1، b، mu و lambda) تنظیم شدند و سپس بر روی یک مجموعه داده آزمایشی (Test Set) با استفاده از معیارهای دقت در ۵ نتیجه اول (P@5)، میانگین رتبه معکوس (MRR) و میانگین دقت متوسط (MAP) ارزیابی شدند. نتایج نشان می‌دهد که مدل BM25 و مدلBigram درونیابی‌شده عملکردی قابل مقایسه داشتند، و مدل BM25 با دستیابی به بالاترین میانگین دقت متوسط (MAP) در مجموعه آزمایشی، بهترین عملکرد را داشت.

۱. مقدمه

هدف اصلی این پروژه ساخت یک موتور جستجو بود که قادر باشد مجموعه‌ای از اسناد را بر اساس ارتباط آن‌ها با پرسش‌های خاص زبان طبیعی رتبه‌بندی کند.

دامنه این پروژه شامل موارد زیر بود:

۱. **پردازش داده‌ها:** دریافت مجموعه‌ای از اسناد متنی و پرسش‌های کاربران.

۲. **نمایه‌سازی (Indexing):** ساخت نمایه‌های معکوس برای امکان بازیابی کارآمد.

۳. **مدل‌سازی:** پیاده‌سازی سه الگوریتم رتبه‌بندی متفاوت (BM25، Unigram LM، Bigram LM).

۴. **ارزیابی:** پیاده‌سازی دستی معیارهای استاندارد IR برای سنجش کمی عملکرد.

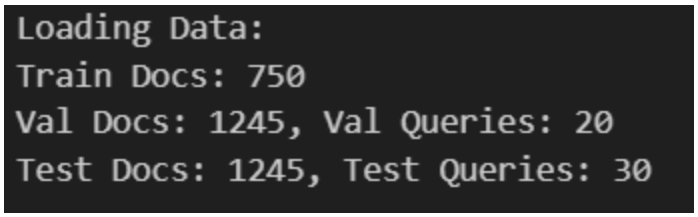
یک محدودیت کلیدی در این پیاده‌سازی، محدودیت واژگان بود. سیستم محدود به حداقل ۵۰۰ کلمه پرتکرار موجود در مجموعه آموزشی بود.

۲. داده‌ها و پیش‌پردازش

آمار مجموعه داده‌ها

مجموعه داده به سه بخش مجزا تقسیم شد تا آموزش و ارزیابی دقیق تضمین شود:

- **مجموعه آموزشی (Training Set):** ۷۵۰ سند. صرفاً برای ساخت واژگان و تخمین احتمال پس‌زمینه (Background Probability) استفاده شد.
- **مجموعه اعتبارسنجی (Validation Set):** ۱۲۴۵ سند و ۲۰ پرسش. برای تنظیم ابرپارامترها استفاده شد.
- **مجموعه آزمایشی (Test Set):** ۱۲۴۵ سند و ۳۰ پرسش. برای ارزیابی نهایی استفاده شد.



شکل ۱

خط لوله پردازش متن

داده‌های متنی خام، بدون ساختار هستند. برای آماده‌سازی داده‌ها جهت نمایه‌سازی، یک خط لوله پیش‌پردازش یکسان بر روی تمام اسناد و پرسش‌ها اعمال شد:

۱. **نرمال‌سازی:** تمام متن‌ها به حروف کوچک تبدیل شدند تا حساسیت به بزرگی و کوچکی حروف از بین برود.

۲. **پاک‌سازی:** علائم نگارشی و کاراکترهای خاص با استفاده از عبارات باقاعده (Regex) حذف شدند.

۳. **توکن‌سازی (Tokenization):** متن بر اساس فضای خالی (Whitespace) به کلمات جداگانه تقسیم شد.

۴. **حذف کلمات توقف (Stopwords):** لیستی از کلمات رایج انگلیسی (مانند"at" ,"is", "the") فیلتر شدند تا نویز کاهش یابد و حجم نمایه کم شود.

```python
def preprocess(text):
    """
    Lowercases, removes punctuation, tokenizes, and removes stopwords.
    """
    if not isinstance(text, str): return []
    text = text.lower()
    # Remove punctuation
    text = re.sub(r'[^a-z0-9\s]', ' ', text)
    tokens = text.split()
    return [t for t in tokens if t not in STOPWORDS]

```

شکل ۲

محدودیت واژگان

پس از پیش‌پردازش **۷۵۰ سند آموزشی**، فرکانس کلمات محاسبه شد و تنها ۵۰۰ کلمه پرتکرار حفظ شدند.

- **تاثیر:** هر کلمه‌ای در اسناد اعتبارسنجی یا آزمایشی (یا پرسش‌ها) که در این لیست ۵۰۰ تایی نبود، در حین نمایه‌سازی و بازیابی حذف شد. این امر به عنوان یک فیلتر شدید انتخاب ویژگی عمل می‌کند اما ساختار معنایی اصلی متن را حفظ می‌کند.

```python
def build_vocab(documents, top_k=500):
    """
    Builds vocabulary from TRAINING documents using top K frequent words.
    Constraint: Length of TF-IDF vector must be at least 500.
    """
    print(f"Building vocabulary from {len(documents)} training documents:")
    all_tokens = []
    for doc_text in documents.values():
        all_tokens.extend(preprocess(doc_text))

    # Count frequencies
    counts = Counter(all_tokens)

    # Select top K (K must be at least 500)
    most_common = counts.most_common(top_k)
    vocab = {term for term, freq in most_common}

    return vocab, counts

# Build Vocab from TRAINING data:
VOCAB, TRAIN_COUNTS = build_vocab(train_passages, top_k=500)
print(f"Vocabulary size: {len(VOCAB)}")

def filter_query(query_text):
    """
    Preprocesses query and filters words not in VOCAB.
    Constraint: New words should not be considered.
    """
    tokens = preprocess(query_text)
    return [t for t in tokens if t in VOCAB]
```

شکل ۳

۳. روش‌شناسی: مدل‌های بازیابی

مدل BM25 (Best Matching 25)

BM25 یک تابع رتبه‌بندی مبتنی بر چارچوب احتمالی ارتباط (Probabilistic Relevance Framework) است. این مدل با معرفی فرکانس کلمه (Term Frequency Saturation) و نرمال‌سازی طول سند، نسبت به TF-IDF استاندارد بهبود یافته است.

فرمول:

امتیاز یک سند D برای یک پرسش Q به صورت زیر محاسبه می‌شود:

$$\text{Score}(D,Q) = \sum_{q \in Q} IDF(q) \cdot \frac{f(q,D) \cdot (k_1 + 1)}{f(q,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

فرمول ۱

که در آن:

- f(q,D) فرکانس کلمه q در سند D است.
- |D| طول سند D است.
- Avgdl میانگین طول اسناد در کل مجموعه است.
- K_1 پارامتری است که اشباع فرکانس کلمه را کنترل می‌کند.
- b پارامتری است که میزان نرمال‌سازی طول را کنترل می‌کند.

پیاده‌سازی:

مؤلفه IDF با استفاده از نسخه احتمالی محاسبه شد:

$$IDF(q) = \log \frac{N - n(q) + 0.5}{n(q) + 0.5} + 1$$

فرمول ۲

```
35        # Precompute IDF
36        for term in self.vocab:
37            df = doc_freqs.get(term, 0)
38            # Standard Probabilistic IDF
39            self.idf[term] = math.log((self.doc_count - df + 0.5) / (df + 0.5) + 1)
```

شکل ۴

این فرمول به صورت دستی پیاده‌سازی شد. فاز تنظیم بر بهینه‌سازی پارامترهای K_1 و b تمرکز داشت.

مدل زبانی تک‌کلمه‌ای (Unigram) با هموارسازی دیریشله

مدل‌سازی زبانی برای بازیابی اطلاعات، اسناد را بر اساس احتمالی که سند D بتواند پرسش Q را تولید کند، رتبه‌بندی می‌کند. مدل تک‌کلمه‌ای فرض استقلال کلمات (Bag-of-Words) را در نظر می‌گیرد.

هموارسازی (Smoothing):

برای حل مشکل احتمال صفر (زمانی که یک کلمه پرسش در سند وجود ندارد)، از هموارسازی دیریشله با استفاده از پارامتر mu استفاده شد. این تکنیک، مدل زبانی سند را با یک مدل پس‌زمینه مجموعه (بر اساس داده‌های آموزشی) ترکیب می‌کند.

فرمول:

$$P(w|D) = \frac{c(w, D) + \mu P(w|C)}{|D| + \mu}$$

فرمول ۳

که در آن:

- c(w,D) تعداد تکرار کلمه w در سند D است.
- P(w|C) احتمال کلمه w در کل مجموعه آموزشی است.
- mu پارامتر هموارسازی است.

```
        for doc_id in candidate_docs:
            doc_len = self.doc_lengths[doc_id]
            log_prob = 0.0

            for term in tokens:
                # Find TF in current doc (naive search in list)
                tf = 0
                for d, c in self.inverted_index[term]:
                    if d == doc_id:
                        tf = c
                        break

                # Dirichlet Smoothing
                p_wc = self.p_C.get(term, 1e-10)
                numerator = tf + (self.mu * p_wc)
                denominator = doc_len + self.mu

                prob = numerator / denominator
                log_prob += math.log(prob)

            scores[doc_id] = log_prob
```

شکل ۵

مدل زبانی دوکلمه‌ای (Bigram) با درونیابی

مدل دوکلمه‌ای با در نظر گرفتن محلی کلمات (جفت‌های کلمات مجاور)، رویکرد تک‌کلمه‌ای را گسترش می‌دهد. این کار عبارات و ساختار محلی را ضبط می‌کند.

فرمول: از آنجا که پراکندگی داده‌ها زیاد است (به‌ویژه با واژگان ۵۰۰ کلمه‌ای)، یک مدل دوکلمه‌ای خالص اغلب غیرقابل اعتماد است. ما یک مدل درونیابی‌شده (Interpolated) را پیاده‌سازی کردیم:

$$P(q_i|q_{i-1}, D) = \lambda P_{Uni}(q_i|D) + (1 - \lambda)P_{Bi}(q_i|q_{i-1}, D)$$

فرمول ۴

که در آن:

- P_{Uni} احتمال هموارشده تک‌کلمه‌ای است.
- P_{Bi} احتمال تجربی ظاهر شدن دوکلمه (Bigram) در سند است.
- lambda وزن درونیابی است (بین ۰.۰ تا ۱.۰).

```
            # 1. First term: P(q1 | D) using Smoothed Unigram
            q1 = tokens[0]
            tf_q1 = doc_seq.count(q1)
            p_uni_q1 = (tf_q1 + self.unigram_model.mu * self.unigram_model.p_C.get(q1, 1e-10)) / (doc_len + self.unigram_model.mu)
            log_prob += math.log(p_uni_q1)

            # 2. Subsequent terms: Bigram P(qi | qi-1, D)
            for i in range(1, len(tokens)):
                qi = tokens[i]
                q_prev = tokens[i-1]

                # Count Bigram C(qi, qi-1, D) and Unigram C(qi-1, D)
                c_bigram = 0
                for j in range(len(doc_seq) - 1):
                    if doc_seq[j] == q_prev and doc_seq[j+1] == qi:
                        c_bigram += 1

                c_prev = doc_seq.count(q_prev)

                # Empirical Bigram Probability
                p_bigram_emp = (c_bigram / c_prev) if c_prev > 0 else 0.0

                # Smoothed Unigram Probability for qi
                tf_qi = doc_seq.count(qi)
                p_uni_smooth = (tf_qi + self.unigram_model.mu * self.unigram_model.p_C.get(qi, 1e-10)) / (doc_len + self.unigram_model.mu)

                # Interpolation
                final_prob = (self.chi * p_bigram_emp) + (self.lam * p_uni_smooth)

                if final_prob <= 0: final_prob = 1e-10
                log_prob += math.log(final_prob)

            scores[doc_id] = log_prob
```

شکل ۶

۴. ارزیابی

برای اندازه‌گیری عینی کیفیت نتایج جستجو، سه معیار استاندارد از پایه پیاده‌سازی شدند:

دقت در K (P@5):

نسبت اسناد بازیابی شده مرتبط در ۵ نتیجه اول را اندازه می‌گیرد.

فرمول :

$$P@K = \frac{\text{Number of Relevant Documents in the Top K}}{K}$$

فرمول ۵

```python
    # 1. Precision at K (P@5)
    retrieved_k = retrieved_doc_ids[:k]
    hits_k = sum(1 for doc in retrieved_k if doc in relevant_set)
    p_at_k = hits_k / k
```

شکل ۷

میانگین رتبه معکوس (MRR):

موقعیت اولین سند مرتبط را ارزیابی می‌کند. این معیار سیستم‌هایی را که اولین پاسخ صحیح را در پایین لیست قرار می‌دهند، جریمه می‌کند.

فرمول :

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i}$$

فرمول ۶

```python
    # 2. Mean Reciprocal Rank (MRR)
    mrr = 0.0
    for i, doc in enumerate(retrieved_doc_ids):
        if doc in relevant_set:
            mrr = 1.0 / (i + 1)
            break
```

شکل ۸

میانگین دقت متوسط (MAP):

یک معیار واحد از کیفیت را در تمام سطوح فراخوانی (Recall) ارائه می‌دهد. این میانگین نمرات دقت متوسط (AP)برای هر پرسش است.

فرمول:

$$MAP = \frac{1}{|Q|} \sum_{q=1}^{|Q|} AP(q)$$

فرمول ۷

```
    # 3. Average Precision (AP)
    # Constraint: Denominator = number of relevant documents
    hits = 0
    sum_prec = 0.0
    for i, doc in enumerate(retrieved_doc_ids):
        if doc in relevant_set:
            hits += 1
            sum_prec += hits / (i + 1)

    if len(relevant_set) > 0:
        ap = sum_prec / len(relevant_set)
    else:
        ap = 0.0

    return p_at_k, mrr, ap
```

شکل ۹

نکته محدودیت: در محاسبهAP ، مخرج کسر به درستی تعداد کل اسناد مرتبط در نظر گرفته شد تا حتی زمانی که سیستم موفق به بازیابی تمام اسناد مرتبط نمی‌شود، معیار دقیق باشد.

۵. نتایج تجربی

فاز ۱: تنظیم ابرپارامترها (مجموعه اعتبارسنجی)

یک جستجوی شبکه‌ای (Grid Search) بر روی مجموعه اعتبارسنجی انجام دادیم تا پارامترهای بهینه برای هر مدل را تعیین کنیم.

تنظیم BM25

ترکیبات {1.2, 1.5, 2.0} را برای k1 و {0.5, 0.75, 1.0} را برای b آزمایش کردیم.

```python
# A. Tune BM25
# Constraints: Test at least 3 values.
k1_vals = [1.2, 1.5, 2.0]
b_vals = [0.5, 0.75, 1.0]
best_bm25_map = -1
best_bm25_params = (1.5, 0.75)

for k in k1_vals:
    for b in b_vals:
        model = BM25(VOCAB, k1=k, b=b)
        _, _, map_score = evaluate_model(model, val_questions, val_judg, val_passages)
        print(f"BM25 (k1={k}, b={b}) -> MAP: {map_score:.4f}")
        if map_score > best_bm25_map:
            best_bm25_map = map_score
            best_bm25_params = (k, b)

print(f"Best BM25 Params: {best_bm25_params}")
```

شکل ۱۰

```
BM25 (k1=1.2, b=0.5) -> MAP: 0.3199
BM25 (k1=1.2, b=0.75) -> MAP: 0.3210
BM25 (k1=1.2, b=1.0) -> MAP: 0.3158
BM25 (k1=1.5, b=0.5) -> MAP: 0.3163
BM25 (k1=1.5, b=0.75) -> MAP: 0.3211
BM25 (k1=1.5, b=1.0) -> MAP: 0.3177
BM25 (k1=2.0, b=0.5) -> MAP: 0.3151
BM25 (k1=2.0, b=0.75) -> MAP: 0.3175
BM25 (k1=2.0, b=1.0) -> MAP: 0.3103
Best BM25 Params: (1.5, 0.75)
```

شکل ۱۱

مشاهده: مدل نسبتاً پایدار بود، اما مقادیر k1=1.5 و b=0.75 بالاترین MAP برابر با **۰.۳۲۱۱** را به دست آوردند.

تنظیم مدل تک‌کلمه‌ای (Unigram)

پارامتر هموارسازی دیریشله {500, 1000, 2000} را برای mu تغییر دادیم.

```python
# B. Tune Unigram LM
# Constraint: Find appropriate mu.
mu_vals = [500, 1000, 2000]
best_uni_map = -1
best_mu = 1000

for mu in mu_vals:
    model = UnigramLM(VOCAB, TRAIN_COUNTS, mu=mu)
    _, _, map_score = evaluate_model(model, val_questions, val_judg, val_passages)
    print(f"Unigram (mu={mu}) -> MAP: {map_score:.4f}")
    if map_score > best_uni_map:
        best_uni_map = map_score
        best_mu = mu

print(f"Best Unigram Mu: {best_mu}")
```

شکل ۱۲

```
Unigram (mu=500) -> MAP: 0.2934
Unigram (mu=1000) -> MAP: 0.2916
Unigram (mu=2000) -> MAP: 0.2899
Best Unigram Mu: 500
```

شکل ۱۳

مشاهده: مقادیر پایین‌تر mu عملکرد بهتری داشتند، که نشان می‌دهد مدل زبانی خاص سند نسبت به مدل پس‌زمینه مجموعه برای این مجموعه داده خاص قابل اعتمادتر بود و mu=500 انتخاب شد.

تنظیم مدل دوکلمه‌ای(Bigram)

وزن درونیابی lambda را تنظیم کردیم، جایی که lambda نشان‌دهنده وزن مؤلفه Unigram است.

```python
# C. Tune Bigram LM
# Constraint: Find optimal lambda.
lambda_vals = [0.1, 0.5, 0.9]
best_bi_map = -1
best_lam = 0.5

# Note: We reuse the best Mu found above for the unigram component
base_unigram = UnigramLM(VOCAB, TRAIN_COUNTS, mu=best_mu)

for lam in lambda_vals:
    model = BigramLM(VOCAB, TRAIN_COUNTS, base_unigram, lam=lam)
    # Re-indexing handled inside evaluate_model -> model.index_documents
    _, _, map_score = evaluate_model(model, val_questions, val_judg, val_passages)
    print(f"Bigram (lambda={lam}) -> MAP: {map_score:.4f}")
    if map_score > best_bi_map:
        best_bi_map = map_score
        best_lam = lam

print(f"Best Bigram Lambda: {best_lam}")
```

شکل ۱۴

```
Bigram (lambda=0.1) -> MAP: 0.3091
Bigram (lambda=0.5) -> MAP: 0.3091
Bigram (lambda=0.9) -> MAP: 0.3116
Best Bigram Lambda: 0.9
```

شکل ۱۵

مشاهده: بهترین عملکرد با lambda=0.9 به دست آمد. این نشان می‌دهد که مدل به شدت احتمالات Unigram (وزن ۹۰٪) را بر احتمالات Bigram (وزن ۱۰٪) ترجیح داده است. این احتمالاً به دلیل اندازه کوچک واژگان (۵۰۰ کلمه) است که باعث می‌شود رخدادهای دوکلمه‌ای پراکنده و از نظر آماری کم‌اهمیت باشند.

فاز ۲: ارزیابی نهایی (مجموعه آزمایشی)

با استفاده از پارامترهای بهینه یافت شده در فاز ۱ (k1=1.5, b=0.75, mu=500, lambda=0.9) ، مدل‌ها را بر روی مجموعه داده آزمایشی (دیده نشده) ارزیابی کردیم.

```python
print("\n--- Phase 2: Final Evaluation on Test Dataset ---")
# Constraint: Evaluate on Test using best params.

# 1. BM25 Final
bm25_final = BM25(VOCAB, k1=best_bm25_params[0], b=best_bm25_params[1])
p5, mrr, map_ = evaluate_model(bm25_final, test_questions, test_judg, test_passages)
print(f"BM25 TEST RESULTS -> P@5: {p5:.4f}, MRR: {mrr:.4f}, MAP: {map_:.4f}")

# 2. Unigram Final
uni_final = UnigramLM(VOCAB, TRAIN_COUNTS, mu=best_mu)
p5, mrr, map_ = evaluate_model(uni_final, test_questions, test_judg, test_passages)
print(f"Unigram TEST RESULTS -> P@5: {p5:.4f}, MRR: {mrr:.4f}, MAP: {map_:.4f}")

# 3. Bigram Final
base_uni_final = UnigramLM(VOCAB, TRAIN_COUNTS, mu=best_mu)
bi_final = BigramLM(VOCAB, TRAIN_COUNTS, base_uni_final, lam=best_lam)
p5, mrr, map_ = evaluate_model(bi_final, test_questions, test_judg, test_passages)
print(f"Bigram TEST RESULTS -> P@5: {p5:.4f}, MRR: {mrr:.4f}, MAP: {map_:.4f}")
```

شکل ۱۶

```
--- Phase 2: Final Evaluation on Test Dataset ---
BM25 TEST RESULTS -> P@5: 0.2800, MRR: 0.4648, MAP: 0.2198
Unigram TEST RESULTS -> P@5: 0.2667, MRR: 0.3744, MAP: 0.1927
Bigram TEST RESULTS -> P@5: 0.2933, MRR: 0.4295, MAP: 0.2156
```

شکل ۱۷

۶. بحث و تحلیل

مقایسه مدل‌ها

نتایج یک سلسله مراتب واضح در عملکرد را نشان می‌دهند:

۱. **BM25** به طور کلی قوی‌ترین مدل بود و بالاترین MRR و MAP را کسب کرد. توانایی آن در اشباع فرکانس کلمه (جلوگیری از اینکه سندی با ۱۰۰ تکرار یک کلمه، ۱۰۰ برابر بالاتر از سندی با ۱ تکرار امتیاز بگیرد) به آن برتری داد.

۲. **Bigram LM** در معیار MAP بسیار شبیه به BM25 عمل کرد (۰.۲۱۵۶ در مقابل ۰.۲۱۹۸) و در معیار Precision@5 حتی از BM25 بهتر بود (۰.۲۹۳۳). این نشان می‌دهد که برای رتبه‌های بسیار بالا (۵ نتیجه اول)، در نظر گرفتن بافت محلی کلمات به یافتن اسناد مرتبط کمک کرده است، حتی اگر رتبه‌بندی کلی (MAP) کمی پایین‌تر باشد.

۳. **Unigram LM** از هر دو مدل دیگر عقب ماند. بدون نرمال‌سازی طول (مانند BM25) یا آگاهی از بافت (مانند Bigram)، این مدل در تمایز بین تطابق‌های باکیفیت و اسنادی که صرفاً کلمات پرسش را زیاد تکرار کرده بودند، مشکل داشت.

تنگنای واژگان

محدودیت استفاده از تنها ۵۰۰ کلمه برتر از مجموعه آموزشی احتمالاً سقف عملکرد همه مدل‌ها را فشرده کرده است.

- **پراکندگی(Sparsity)** : اصطلاحات خاص پرسش (مانند اسامی خاص یا اصطلاحات فنی) اگر در ۵۰۰ کلمه برتر نبودند، احتمالاً در طول پیش‌پردازش فیلتر شده‌اند.
- **شکستBigram** : ترجیح برای lambda ۰.۹ در مدل Bigram این موضوع را تایید می‌کند. با چنین واژگان کوچکی، دوکلمه‌ای‌های معتبر (مثلاً"new york") ممکن است یک یا هر دو کلمه را از دست داده باشند، که مؤلفه Bigram را بی‌استفاده می‌کند. مدل "یاد گرفت" که مؤلفه Bigram را نادیده بگیرد تا دقت را به حداکثر برساند.

شکاف تعمیم (Generalization Gap)

افت عملکرد قابل توجهی بین مجموعه اعتبارسنجی (حدود MAP۰.۳۲) و مجموعه آزمایشی (حدود ۰.۲۲ MAP) وجود داشت. این افت ۳۰ درصدی دو احتمال را نشان می‌دهد:

۱. **عدم تطابق داده‌ها** :پرسش‌های تست ممکن است ذاتاً دشوارتر بوده یا بر موضوعاتی متمرکز بوده‌اند که در واژگان ۵۰۰ تایی کمتر نمایندگی شده‌اند.

۲. **بیش‌برازش(Overfitting)** : پارامترهای تنظیم شده بر روی مجموعه اعتبارسنجی (به ویژه lambda درBigram) ممکن است مختص ویژگی‌های پرسش‌های اعتبارسنجی بوده باشند.

نتیجه گیری

در این پروژه، سه مدل بنیادی بازیابی اطلاعات را تحت محدودیت‌های سخت پیاده‌سازی و ارزیابی کردیم. **مدل BM25** ثابت کرد که موثرترین و قابل‌تعمیم‌ترین رویکرد برای این مجموعه داده است، زیرا فرکانس کلمه و طول سند را به طور موثر متعادل می‌کند. **مدل زبانی Bigram**، با وجود اینکه در P@5 خوب بود، از پراکندگی داده‌های ناشی از محدودیت واژگان رنج برد و مجبور شد رفتاری تقریباً مشابه با مدل Unigram داشته باشد.