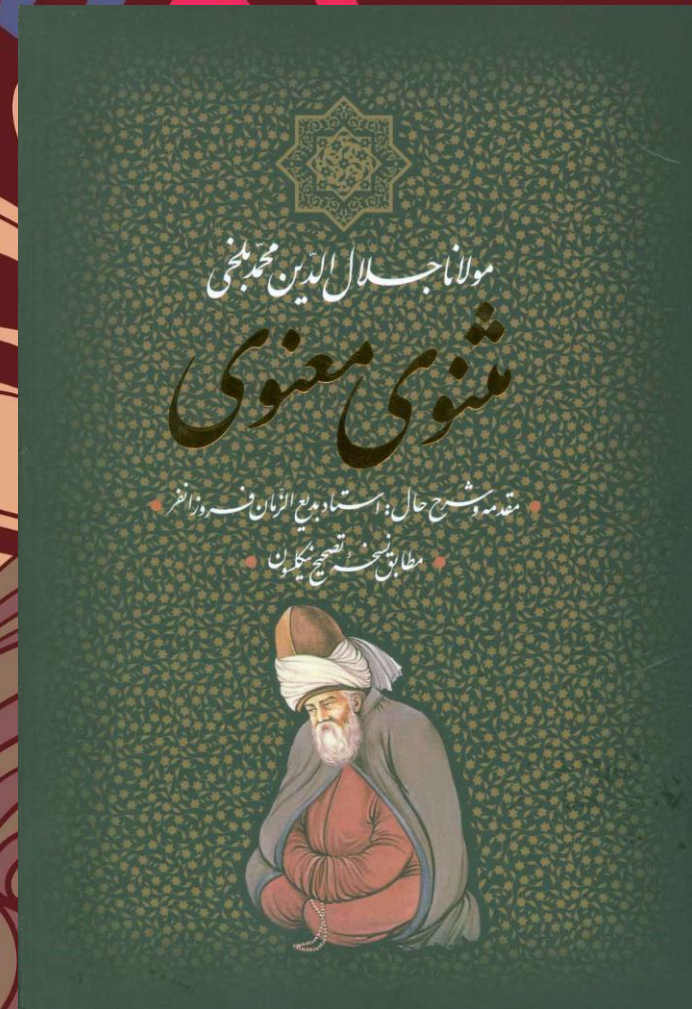


تمرین دوم
استفراج عبارات کلیدی

شش دفتر مولوی



درباره‌ی متن

مثنوی معنوی یکی از مهم‌ترین آثار عرفانی و ادبی قرن هفتم هجری شمسی است که برخی به آن لقب مصحف ثانی داده‌اند. گویی مثنوی قرآن دوم ماست. عبدالرحمن **جامی** در توصیف این اثر عرفانی ارزشمند آن را قرآن در زبان فارسی نامیده و جلال‌الدین همایی شاعر و نویسنده معاصر ایرانی نیز آن را هم سروده‌های گاتا و عهد جدید دانسته است. بسیاری از اهل طریقت، مثنوی معنوی را کتابی برای دستیابی به حقیقت می‌دانند.

نمونه متن

- | | | |
|-------|--------------------------------|--------------------------------|
| 11.1 | خوش نوا و سبز و گویا طوطیی | بود بقالی و او را طوطیی |
| 11.2 | نکته گفتی با همه سوداگران | بر دکان بودی نگهبان دکان |
| 11.3 | در نوای طوطیان حاذق بدی | در خطاب آدمی ناطق بدی |
| 11.4 | بر دکان طوطی نگهبانی نمود | خواجه روزی سوی خانه رفته بود * |
| 11.5 | بهر موشی، طوطیک از بیم جان | گربه ای بر جست ناگه از دکان * |
| 11.6 | شیشه های روغن گل را بریخت | جست از صدر دکان، سویی گریخت |
| 11.7 | بر دکان بنشست فارغ خواجه وش | از سوی خانه پیامد خواجه اش |
| 11.8 | بر سرش زد، گشت طوطی کل ز ضرب | دید پُر روغن دکان و جاش چرب |
| 11.9 | مرد بقال از ندامت آه کرد | روزگی چندی سخن کوتاه کرد |
| 11.10 | کافتاب نعمتم شد زیر میغ | ریش بر میکند و میگفت: ای دریغ |
| 11.11 | که زدم من بر سر آن خوش زبان | دست من بشکسته بودی آن زمان |
| 11.12 | تا بیاید نطق مرغ خویش را | هدیه ها میداد هر درویش را |
| 11.13 | بر دکان بنشسته بُد نومیدوار | بعد سه روز و سه شب حیران و زار |
| 11.14 | کای عجب، این مرغ کی آید بگفت ؟ | با هزاران غصه و غم گشته جفت * |
| 11.15 | واز تعجب، لب بدندان میگرفت | مینمود آن مرغ را هر گون شگفت * |

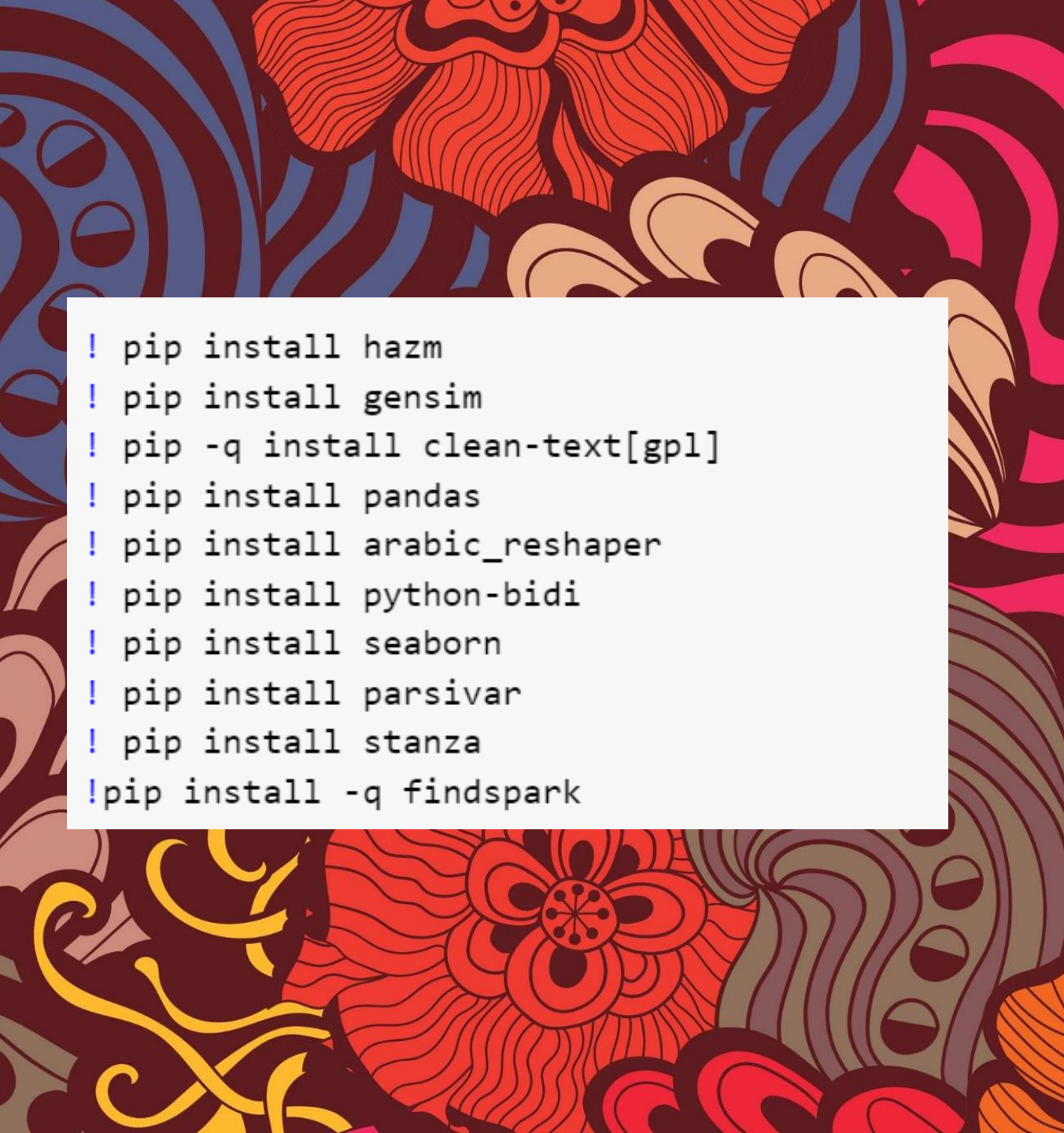
خوش‌نوایی سبز و گویا طوطیی
نکته گفتی با همه سوداگرا
در نوای طوطیان حاذق بدی
بر دکان طوطی نگهبانی نمود
بهر موشی طوطیک از بیم جان
شیشه‌های روغن گل را بریخت

بود بقالی و وی را طوطیی
بر دکان بودی نگهبان دکان
در خطاب آدمی ناطق بدی
خواجه روزی سوی خانه رفته بود
گربه‌ای برجست ناگه بر دکان
جست از سوی دکان سویی گریخت



چالش‌های متن

- وجود انواع مختلف علائم نگارشی و سمبل‌ها در متن جمع‌آوری شده
- وجود انواع قدیمی فعل نظیر همی‌برد - بردمی
- وجود نیم‌فاصله که موجب ابهام در شناخت واژه "می" به معنی باده و "می" پیشوند فعل مضارع می‌شود.
- وجود حروف الفبای عربی که در زبان الفبای زبان فارسی وجود ندارند.
- ترکیب متون عربی و فارسی در شعر
- در هم‌ریختگی اجزای جمله به سبب حفظ وزن و قافیه در شعر
- وجود کلمات توقف که در فارسی روزمره رایج نیستند نظیر "کاندر"



```
! pip install hazm
! pip install gensim
! pip -q install clean-text[gpl]
! pip install pandas
! pip install arabic_reshaper
! pip install python-bidi
! pip install seaborn
! pip install parsivar
! pip install stanza
!pip install -q findspark
```

کتابخانه های استفاده شده

Plot_top_words

این تابع مسئول رسم نمودار های
مربوط به مدل سازی
موضوعی (topic modeling)
است.

```
def plot_top_words(model, feature_names, n_top_words, title):  
    fig, axes = plt.subplots(2, 5, figsize=(30, 15), sharex=True)  
    axes = axes.flatten()  
    for topic_idx, topic in enumerate(model.components_):  
        top_features_ind = topic.argsort()[::-n_top_words - 1 : -1]  
        top_features = [feature_names[i] for i in top_features_ind]  
        weights = topic[top_features_ind]  
  
        ax = axes[topic_idx]  
        ax.barh(top_features, weights, height=0.7)  
        ax.set_title(f"Topic {topic_idx + 1}", fontdict={"fontsize": 30})  
        ax.invert_yaxis()  
        ax.tick_params(axis="both", which="major", labelsize=20)  
        for i in "top right left".split():  
            ax.spines[i].set_visible(False)  
        fig.suptitle(title, fontsize=40)  
  
    plt.subplots_adjust(top=0.90, bottom=0.05, wspace=0.90, hspace=0.3)  
    plt.show()
```

Make_farsi_text

معمولا در صورت فارسی بودن
برچسبها در نمودارهای
matplotlib به درستی نمایش
داده نمی شوند؛ با این استفاده از این
تابع نمایش برچسبها اصلاح شده
است.

```
[8] from arabic_resaper import reshape
    from bidi.algorithm import get_display
    import seaborn as sns

[9] def make_farsi_text(x):
    if x == 'الله':
        return 'هلا'
    reshaped_text = reshape(x)
    farsi_text = get_display(reshaped_text)
    return farsi_text
```


Clean_text

این تابع یک متن را گرفته و با
حذف موارد ناخواسته نظیر stop
words ها و سایر سمبل‌ها آن را
تمییز می‌کند.


```
[20] def clean_text(text):  
    text = re.sub(spaces_reg, " ", text)  
    text = re.sub(symbols_reg, "", text)  
    text = remove_arabic(text)  
    text = normalizer.normalize(text)  
    regex = r"\b(?:" + "|".join(map(re.escape, stopwords)) + r")\b"  
    text = re.sub(regex, " ", text)  
    text = re.sub(spaces_reg, " ", text)  
    text = re.sub("(\s)+", " ", text)  
    text = text.strip()  
    return text
```


Process_couplet

تابع مقابل یک بیت را گرفته و
شماره شعر در دفتر، شماره بیت،
بیت، بیت تمییز شده و مصرع اول
و دوم تمییز شده را در پاسخ باز
می‌گرداند.




```
[21] def process_couplet(text):
    result = re.search("(\\d{1,3})\\.\\.\\d{1,3})", text)
    # check if line contains a couplet
    if result:
        pno, cno = result.groups()
        # delete Daftar and Poem number
        couplet = re.sub("(\\d{1,3})\\.\\.\\d{1,3})", "", text)
        # extract mesra
        hemistich = couplet.split("\\t")[1:3]
        # extract important tokens in mesra
        cleaned_hemistich = [clean_text(h) for h in hemistich]
        # clean spaces
        couplet = re.sub(spaces_reg, " ", couplet)
        return pno, cno, couplet, " ".join(cleaned_hemistich), cleaned_hemistich[0], cleaned_hemistich[1]
    return None
```



تکه کد مقابل دفاتر را به تابع
مسئول تمییز کردن ابیات
می‌دهد و در یک لیست ابیات
تمییز شده را ذخیره می‌کند.

```
daftar = 0
for couplet in masnavi_file:
    if re.search("(^(:دفتر).*(:مثنوی)$", couplet):
        daftar += 1
        if daftar == 7:
            print("Processing Completed!")
            break
        print(f"Processing Daftar {daftar}")
    else:
        process_result = process_couplet(couplet)
        if process_result:
            pno = process_result[0]
            cno = process_result[1]
            c = process_result[2]
            cc = process_result[3]
            h1 = process_result[4]
            h2 = process_result[5]
            data = (daftar, pno, cno, c, cc, h1, h2)
            masnavi.append(data)
```


در این یک تابع کاربر اسپارک
تعریف شده تا یک ستون شامل
ابیات tokenized شده با
کتابخانه‌ی parsivar را
ذخیره کند.



```
[42] from pyspark.sql.types import ArrayType, StringType
```

```
[43] token_udf = udf(  
    lambda couplet: pars_tokenizer.tokenize_words(couplet),  
    StringType())
```

```
[44] masnavi_pdf = masnavi_pdf.withColumn("ParsTokens", token_udf(masnavi_pdf.Couplet))
```



```
[42] from pyspark.sql.types import ArrayType, StringType
```

```
[43] token_udf = udf(  
    lambda couplet: pars_tokenizer.tokenize_words(couplet),  
    StringType())
```

```
[44] masnavi_pdf = masnavi_pdf.withColumn("ParsTokens", token_udf(masnavi_pdf.Couplet))
```




در ادامه ی این کد سعی داشتم که با استفاده از روشی مشابه آنچه برای tokenize کردن ابیات با استفاده از spark انجام شده، عملیات pos tagging را بر روی ابیات انجام دهم، زیرا انجام این پردازش به صورت معمول نزدیک به ۱۱ ساعت به طول می انجامید.

اما چون wrapper نوشته شده در parsivar از pointer استفاده می کرد، قابل استفاده در pyspark نبود.

در نتیجه پس از مطالعه زیاد نتیجه گرفتم به سراغ مدل Stanford نوشته شده به نام stanza بروم که از زبان فارسی نیز به خوبی پشتیبانی می کند.

یک pipeline سنتی nlp را
کانفیگ کرده ام تا به ترتیب
tokenize ، multi word
tokenize و pos
tagging انجام شود.




```
[48] nlp = stanza.Pipeline(lang='fa', processors='tokenize,mwt,pos')
```


```
2021-12-13 13:40:15 INFO: Loading these models for language: fa (Persian):
```

```
=====
| Processor | Package |
|-----|-----|
| tokenize | perdt   |
| mwt      | perdt   |
| pos      | perdt   |
=====
```

```
2021-12-13 13:40:15 INFO: Use device: cpu
2021-12-13 13:40:15 INFO: Loading: tokenize
2021-12-13 13:40:15 INFO: Loading: mwt
2021-12-13 13:40:15 INFO: Loading: pos
2021-12-13 13:40:15 INFO: Done loading processors!
```




یک کلاس نوشته شده تا
document های pos tag
شده را در آن ذخیره کنم.



```
class PosTag:
    def __init__(self, id, text, upos, xpos, feats, start_char, end_char):
        self.id = id
        self.text = text
        self.upos = upos
        self.xpos = xpos
        self.feats = {x.split("=")[0]:x.split("=")[1] for x in [x for x in feats.split("|")]} if feats else {}
        self.start_char = start_char
        self.end_char = end_char

    def to_dict(self):
        return {
            'id': self.id,
            'text': self.text,
            'upos': self.upos,
            'xpos': self.xpos,
            'feats': self.feats,
            'start_char': self.start_char,
            'end_char': self.end_char
        }
```




یک کلاس نوشته شده تا
document های pos tag
شده را در آن ذخیره کنم.




```
class PosTag:
    def __init__(self, id, text, upos, xpos, feats, start_char, end_char):
        self.id = id
        self.text = text
        self.upos = upos
        self.xpos = xpos
        self.feats = {x.split("=")[0]:x.split("=")[1] for x in [x for x in feats.split("|")]} if feats else {}
        self.start_char = start_char
        self.end_char = end_char

    def to_dict(self):
        return {
            'id': self.id,
            'text': self.text,
            'upos': self.upos,
            'xpos': self.xpos,
            'feats': self.feats,
            'start_char': self.start_char,
            'end_char': self.end_char
        }
```

کلاس مقابل توابع ساده ای نظیر
پیدا کردن فعل، اسم و صفت های
رایج در مثنوی را انجام می دهد.



```
class MasnaviPosBasedAnalyze:
    def __init__(self, pos_tags):
        self.masnavi_tags = list()
        for i, tags in enumerate(pos_tags):
            try:
                sent = tags.sentences[0]
                for word in sent.words:
                    self.masnavi_tags.append(PosTag(
                        i,
                        word.text,
                        word.upos,
                        word.xpos,
                        word.feats,
                        word.start_char,
                        word.end_char))
            except:
                pass
        self.masnavi_pos_df = pd.DataFrame.from_records([t.to_dict() for t in self.masnavi_tags])
```



اطلاعات کلی در مورد دیتاست شش دفتر مولوی

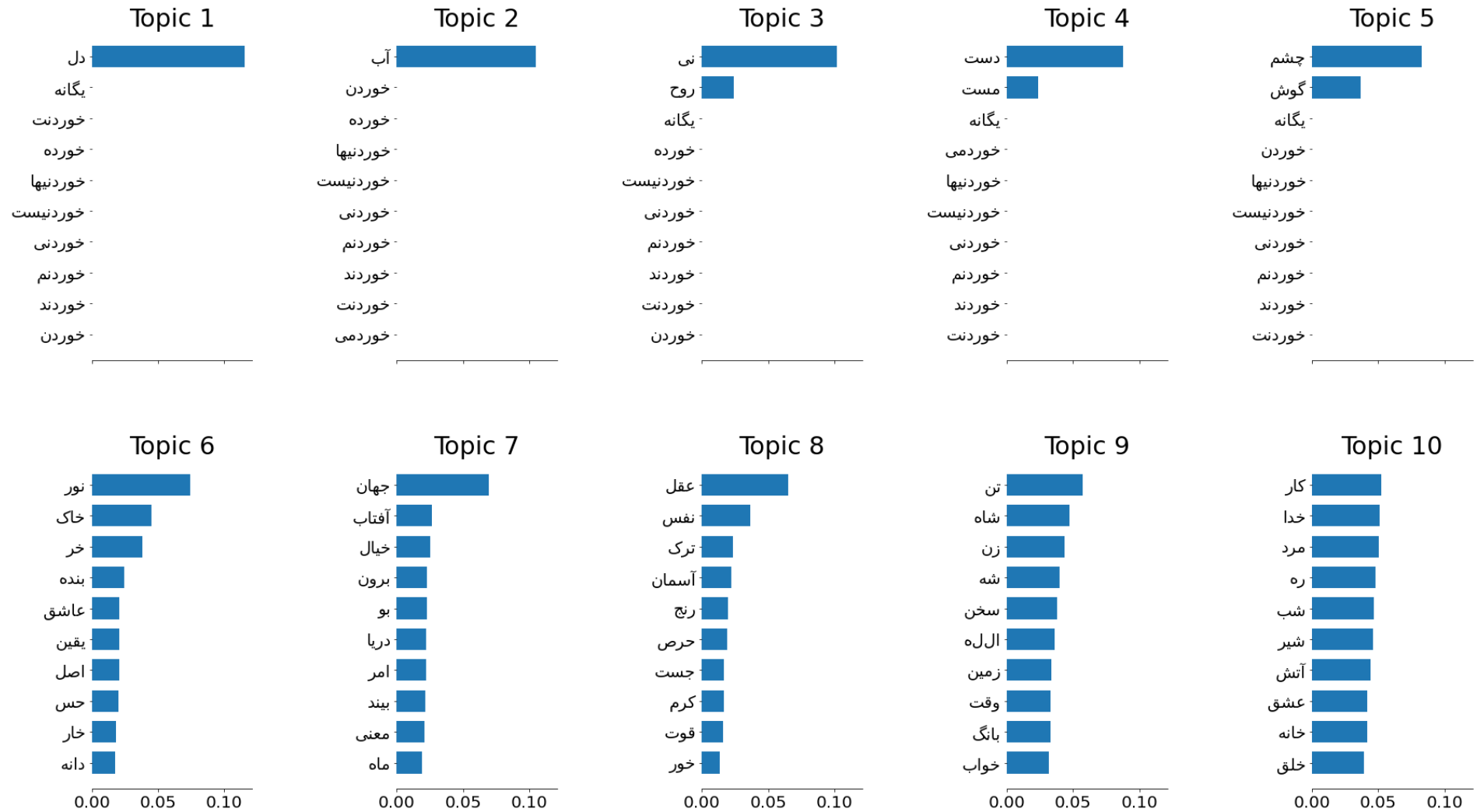
```
▶ print ('%-16s' % 'Number of words', '%-16s' % len(all_words))  
print ('%-16s' % 'Number of unique words', '%-16s' % len(set(all_words)))  
avg=np.sum([len(word) for word in all_words])/len(all_words)  
print ('%-16s' % 'Average word length', '%-16s' % avg)  
print ('%-16s' % 'Longest word', '%-16s' % all_words[np.argmax([len(word) for word in all_words])])
```

```
↳ Number of words 162712  
Number of unique words 22055  
Average word length 4.006084369929692  
Longest word استخوانهاشان
```

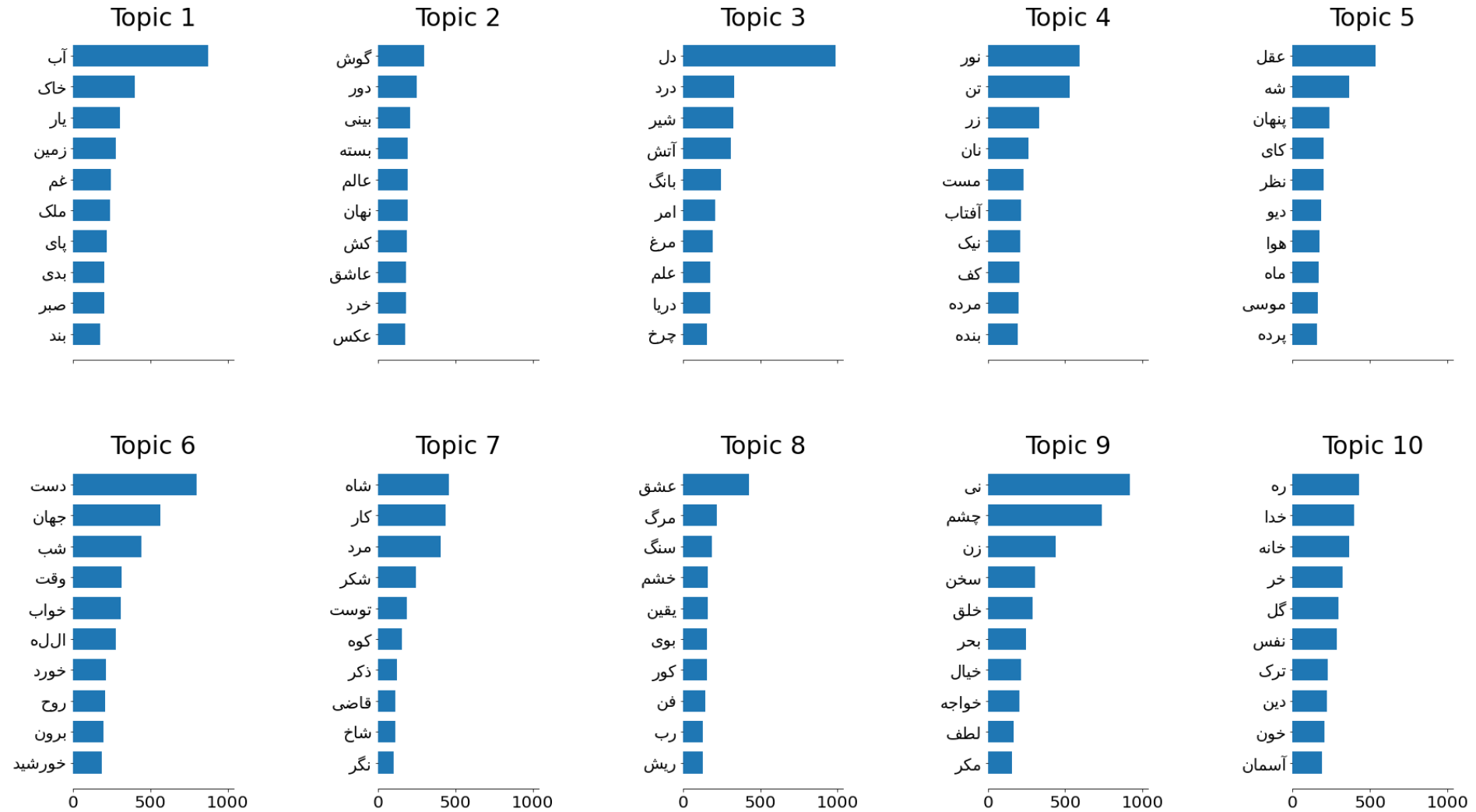


دو اسلاید بعدی خروجی TOPIC MODELING انجام شده در شش دفتر در الگوریتم‌های LDA و NMF را نشان می‌دهد.

Topics in NMF model



Topics in LDA model





اسلاید بعد فراوانی کلمات رایج مورد استفاده به تفکیک شش دفتر را
نشان می‌دهد.

