CHUQUR O‘QITISH USULLARI ASOSIDA MATNLI MA’LUMOTLARNING QISQACHA MAZMUNINI CHIQARISH METODLARI

Toshkent xalqaro moliyaviy boshqaruv va texnologiyalar universiuteti
“Arxitektura va raqamli texnologiyalari”
kafedrasi katta o‘qituvchisi ***F.N.Iskandarova***
feruza621988@gmail.com

Annotatsiya: Ushbu ishda chuqur o‘qitish usullari asosida matnli ma’lumotlarning qisqacha mazmunini avtomatik tarzda chiqarish algoritmini ishlab chiqish masalasi ko‘rib chiqilgan. Bugungi kunda katta hajmdagi matnli axborotni qisqa, aniq va mazmunli shaklga keltirishga bo‘lgan ehtiyoj ortib bormoqda. Ayniqsa, yangiliklar, ilmiy maqolalar, hujjatlar va boshqa matnli manbalarni tezkor tahlil qilishda bunday texnologiyalarning roli katta ahamiyat kasb etadi. Ish natijasida matnlarning mazmunini tushunarli va ixcham shaklda chiqarib bera oladigan algoritm ishlab chiqildi.

Kalit so’zlat: *Frequency-Based Extractive Summarization, Heuristik algoritm, Statistik yondashuv, TextRank, SumBasic, BERT, T5, Transformer modellar.*

Kirish: Raqamli texnologiyalar rivojlanishi bilan matnli ma’lumotlarni avtomatik tarzda tahlil qilish va ularning qisqacha mazmunini aniqlash tobora muhim ahamiyat kasb etmoqda. Bugungi kunda sun’iy intellekt, axborot izlash tizimlari, kompyuter lingvistikasi va ommaviy axborot vositalarida katta hajmdagi matnlarni samarali qayta ishlash talab etiladi. Katta hajmdagi va murakkab tuzilgan matnlar foydalanuvchilar tomonidan to‘liq o‘rganilishi vaqt talab qilgani sababli, ularning mazmunini aniq, ixcham va tushunarli tarzda ifodalovchi algoritmlarga ehtiyoj ortib bormoqda. Noaniq yoki ortiqcha ma’lumotlar matnning noto‘g‘ri talqin qilinishiga, axborotga asoslangan qarorlar sifatining pasayishiga olib kelishi mumkin. Shu sababli, matnli ma’lumotlarning qisqacha mazmunini chiqarish usullarini ishlab chiqish va ularning samaradorligini baholash bugungi kunda dolzarb ilmiy-texnikaviy masalalardan biri hisoblanadi.

Ishlatilgan model turi: Extractive summarization (chiqarib olishga asoslangan xulosa chiqarish) — bu model matnning eng muhim gaplarini ajratib oladi, ularni tahrirlashsiz natijada ko‘rsatadi.

Model qanday ishlaydi (asosiy qismlar):

1. Gaplarga ajratish:

```
gaplar = re.split(r'[.!?\n]+', matn)
```

Matn nuqta, savol belgisi, yoki yangi qatordan gaplarga ajratiladi.

2. So‘zlar chastotasini hisoblash:

sozlar = re.findall(r'\w+', matn.lower())

chastota = Counter(sozlar)

Matndagi barcha so‘zlar olinib, har birining necha marta uchragani (chastotasi) hisoblanadi.

3. Har bir gapga ball berish:

for g in gaplar:

...

ball = sum(chastota[s] for s in sozlar_g)

Gapdagi so‘zlar chastotasining yig‘indisi orqali har bir gapga ball beriladi — bu ball gapning muhimligini ifodalaydi.

4. Eng baland ball olgan gaplarni tanlash:

tanlangan_gaplar = [g for _, g in sorted(ballar, reverse=True)[:gap_soni]]

Eng baland ball to‘plagan (ya'ni, muhim deb baholangan) gap_soni ta gap tanlanadi (default = 3).

Model turi: Extractive summarization (statistikaga asoslangan)

Asosiy komponentlar:

re – matnni gap va so‘zlarga ajratish uchun

Counter – so‘z chastotasini hisoblash

sorted() – eng muhim gaplarni tanlash

Chuqur o‘rganish yoki til modellari ishlatilmagan – bu model soddaligi bilan ajralib turadi, tez va oson ishlaydi, lekin semantik tahlil (mazmuniy tushunish) qilmaydi.

Agar siz semantik tahlil qiladigan model (masalan, BERT, GPT, T5) ishlatmoqchi bo‘lsangiz, bu uchun NLP kutubxonalar (HuggingFace Transformers) va model yuklash (GPU/CPUda) zarur bo‘ladi.

Quyida shu algoritm haqida batafsil tushuntiraman:

Ishlatilgan algoritm: So‘z chastotasi asosidagi xulosa chiqarish algoritmi

Bu algoritm murakkab matematik modellar yoki neyron tarmoqlarsiz ishlaydi. Uning ish tamoyili oddiy statistikaga asoslangan. Keling, bu algoritmni bosqichma-bosqich ko‘rib chiqamiz:

1. Matnni gaplarga ajratish

gaplar = re.split(r'[.!?\n]+', matn)

Matnning har bir gapini aniqlash uchun . ! ? yoki \n belgilariga ajratiladi.

Bu — gap segmentatsiyasi algoritmining oddiy ko‘rinishi.

2. Matndagi barcha so‘zlarni chiqarib olish va chastotasini hisoblash

sozlar = re.findall(r'\w+', matn.lower())

chastota = Counter(sozlar)

Har bir so‘z aniqlanadi (\w+ – harf va raqamlardan iborat ketma-ketlik).

Har bir so‘z necha marta takrorlanganligi hisoblanadi.

Bu bosqichda so‘z chastotasi aniqlanadi — bu statistik algoritmda muhim o‘rin tutadi.

3. Har bir gapga ball berish

ball = sum(chastota[s] for s in sozlar_g)

Har bir gap uchun uning ichidagi so‘zlarning chastotalari yig‘indisi hisoblanadi.

Yig‘indi qancha katta bo‘lsa, demak, bu gap matnning ko‘p ishlatilgan muhim so‘zlarini o‘z ichiga oladi.

Bu yerda asosiy algoritmik g‘oya shuki: matndagi muhim so‘zlar ko‘p ishlatiladi, shuning uchun ular joylashgan gaplar ham muhim bo‘ladi.

4. Eng katta ball olgan gaplarni tanlash

sorted(ballar, reverse=True)[:gap_soni]

Har bir gapning balini kamayish tartibida saralaydi.

Eng yuqori ball olgan N ta gap tanlanadi (gap_soni – odatda 3).

Bu gaplar birlashtirib, qisqacha xulosa shakllantiriladi.

Bu heuristik asoslangan (qoida asosidagi) oddiy "Frequency-Based Extractive Summarization" algoritmidir. Bu ba'zi hollarda "TextRank" yoki "SumBasic" kabi algoritmlarning soddalashtirilgan shakli hisoblanadi, lekin sizning holatda manual scoring ya'ni oddiy ball berish orqali amalga oshirilgan.

Afzalliklari:

- Juda tez ishlaydi
- Har qanday til uchun (shu jumladan o‘zbek tili) oddiy sozlash bilan mos keladi
- Matnni tahlil qilish uchun sun’iy intellekt talab qilmaydi

Kamchiliklari:

- So‘zlarning ma’nosini tushunmaydi
- Sinonimlar, ironiyalar, kontekstual muhim gaplarni tushunmasligi mumkin
- Semantikani hisobga olmaydi

Agar siz bu oddiy algoritmni yanada kuchliroq sun’iy intellekt bilan almashtirmoqchi bo‘lsangiz, masalan, TextRank, BERTSum, yoki T5 kabi modellarni ishlatish mumkin. Lekin ular uchun NLP kutubxonalari (masalan, HuggingFace Transformers) va kuchli hisoblash quvvatiga ega tizim kerak bo‘ladi.

Heuristik asoslangan (qoida asosidagi) algoritmlar nima?

Heuristika — bu muammoni yechish uchun tezkor, tajribaga asoslangan, ammo har doim to‘g‘ri yoki optimal natija bermaydigan usullardir.

Oddiy ta’rifi bilan: heuristika — bu to‘liq tahlil qilmasdan, yaxshi yechimni tez topish uchun ishlatiladigan qoida yoki usul.

1-jadval

Heuristik algoritmlarning xususiyatlari.

Xususiyat	Tavsifi
Oddiylik	Heuristikalar ko‘pincha murakkab matematik modellarni talab qilmaydi.
Tezlik	Hisoblash jihatdan yengil, tez ishlaydi.
Yaxshi yechim	Har doim mukammal emas, lekin "yetarlicha yaxshi" natijani beradi.
Qoida asosida ishlaydi	Belgilangan qoidalar yoki mezonlarga asoslangan qarorlar qabul qiladi.

Matnni qisqacha mazmunlashda heuristik algoritmlar qanday ishlaydi?

Matndan xulosa olishda heuristik usullar ma’lum qoidalarga asoslangan holda gaplarning ahamiyatini baholaydi. Eng oddiy, siz foydalanayotgan modeldagi kabi:

Asosiy heuristik qoidalar:

- So‘z chastotasi qoidasi (Word Frequency Rule):
- Matnda ko‘p uchragan so‘zlar muhim hisoblanadi.
- Chastota qancha yuqori bo‘lsa — so‘z shuncha muhim.

Gap uzunligi qoidasi (Optional):

- Juda qisqa yoki juda uzun gaplar e'tiborga olinmaydi (g‘oyat muhim bo‘lmasligi mumkin).

Biror muhim joyda turgan gap qoidasi (Position Rule):

- Birinchi yoki oxirgi gaplar odatda matnning asosiy g‘oyasini bildiradi. (Ko‘plab algoritmlarda bu asosiy qoida sifatida ishlatiladi.)

Gapdagi muhim so‘zlar soni:

- Gapda necha marta muhim deb belgilangan so‘z ishlatilgan — shunga ko‘ra ball beriladi.

Heuristik algoritm ishlash jarayoni:

- Matnni gaplarga bo‘lish
- Har bir gapdagi so‘zlarni ajratish
- So‘z chastotasini hisoblash

Har bir gapga ball berish:

- Bu ball — gapdagi so‘zlarning chastotalari yig‘indisi
- Eng yuqori ball to‘plagan gaplarni tanlash
- Tanlangan gaplarni tartiblab chiqarish (mazmun sifatida)
- Oddiy pseudokod (algoritm ko‘rinishi):

1. Matnni gaplarga ajrat
2. Har bir so‘zning chastotasini hisobla
3. Har bir gapga:
   - So‘zlarining chastotasiga qarab ball ber
4. Eng yuqori ball olgan N ta gapni tanla
5. Ularni tartiblab, yakuniy mazmun sifatida chiqar

2-jadval.

Frequency-Based Extractive Summarization algoritmlarning afzallik xususiyatlari.

Afzallik	Tavsif
Tez ishlaydi	So‘z chastotasini hisoblash oson va tez
Oddiy implementatsiya	Har qanday dasturlash tilida osongina amalga oshiriladi
Sun’iy intellekt talab qilmaydi	Neyron tarmoqlar, model yuklash, katta resurslar kerak emas
Tilga nisbatan mustaqil	Har qanday tilga sozlash mumkin, shartli tahlil talab qilinmaydi

3-jadval

Frequency-Based Extractive Summarization algoritmlarning kamchilik xususiyatlari.

Kamchilik	Tavsif
Semantikani tushunmaydi	So‘zlar ma’nosini yoki kontekstni tushunmaydi
Sinonimlar farqlanmaydi	Masalan, “muammo” va “masala” bir xil deb qaralmaydi
Gaplarni tahrirlamaydi	Faqat original matndagi gaplarni qayta tartiblaydi, qayta yozmaydi
Chalg‘ituvchi gaplar kiritilishi mumkin	Agar muhim bo‘lmagan, ammo ko‘p takrorlangan so‘zlar bo‘lsa, noto‘g‘ri xulosa chiqishi mumkin

Xulosa: Frequency-Based Extractive Summarization — bu statistik yondashuvga asoslangan soddalashtirilgan, samarali algoritm bo‘lib, matnni tahlil qilishda tez, resurs tejamkor va dasturiy jihatdan oson bo‘lgan usul hisoblanadi. U ilmiy, o‘quv yoki amaliy loyihalarda boshlang‘ich bosqich uchun juda foydali vositadir.

Adabiyotlar:

1. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.
2. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems (NeurIPS).
3. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
4. Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21(140), 1–67.
5. Nallapati, R., Zhai, F., & Zhou, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. arXiv preprint arXiv:1602.06023.