EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

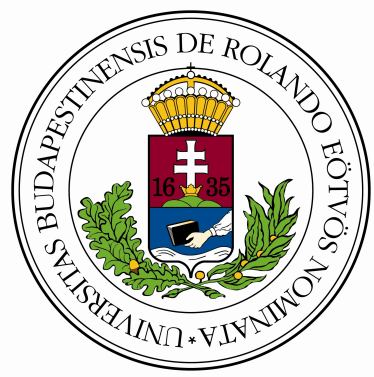
Bartalis Dávid

Szubmoduláris függvények alkalmazása a mesterséges nyelvfeldolgozásban

Témavezető:

Bérczi-Kovács Erika

Operációkutatási Tanszék



Budapest, 2020

Tartalomjegyzék

Köszönetnyilvánítás **2**

Bevezetés **3**

1. Alapfogalmak **4**

1.1. Szubmodularitás 4

1.2. Szupermodularitás, modularitás 6

1.3. Fontos tulajdonságok 7

2. Monoton szubmoduláris függvények maximalizálása **9**

3. Szubmodularitás a szövegfeldolgozásban **11**

3.1. Tömörítés adott felső korláttal 11

3.1.1. Maximális hozzáadott relevancia (MMR) 12

3.1.2. Módosított mohó algoritmus 13

3.1.3. Koncepció alapú tömörítés 18

3.2. Tömörítés adott alsó korláttal 19

3.2.1. Wolsey algoritmusa 19

3.2.2. Minimális domináns halmaz keresése 21

3.3. Automatikus értékelés 22

3.4. További monoton szubmoduláris célfüggvények 23

3.4.1. Az információ mennyiségét mérő függvény 23

3.4.2. A változatosságot mérő függvény 25

4. Apricot - Tanulási teszthalmazok **26**

4.1. Szolgáltató elhelyezési függvény 26

4.2. Tulajdonság-alapú függvény 27

4.3. Példák 28

5. Összegzés **32**

Irodalomjegyzék **33**

Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani témavezetőmnek, Bérczi-Kovács Erikának, aki segítségével hozzájárult a szakdolgozatom elkészítéséhez. Egész idő alatt támogatott és bármikor számíthattam rá, minden egyes kérdésemre készséggel és türelemmel válaszolt.
Továbbá szeretném hálámat kifejezni a családomnak a támogatásukért, illetve a hallgatótársaimnak, barátaimnak a segítségükért.

Bevezetés

Ismert, hogy a szubmoduláris függvények rendkívül nagy szerepet játszanak a matematika több területén is, mint például a gráfelméletben, a valószínűségszámításban, az operációkutatásban, a kombinatorikus optimalizálásban vagy a játékelméletben. Szakdolgozotom célja annak bemutatása, hogy a szubmoduláris függvények nemcsak ezeken a területeken, hanem a mesterséges nyelvfeldolgozási folyamatok során is gyakran megjelennek.

Az első fejezetben néhány alapfogalmat definiálok, alapvető állításokat, tételeket ismertetek, melyeket a további fejezetekben segítségül hívok. Ezek után a szubmoduláris maximalizálásról, annak bonyolultságáról esik szó, majd megmutatom, hogy a különböző lényegkiemelési feladatokhoz milyen szubmoduláris modelleket célszerű használni, és ezek maximalizálásához milyen közelítő algoritmusokat lehet alkalmazni. Végül egy olyan Python csomagot ismertetek, ami szubmoduláris függvények segítségével lényegkiemelési feladatokat old meg, ezáltal gyakran használják a gépi tanulási folyamatok lerövidítésére. Ezen program segítségével néhány példán keresztül szemléltetni fogom az egyes szubmoduláris függvények hatékonyságát.

1. Alapfogalmak

1.1. Szubmodularitás

1.1.1. Definíció (Szubmoduláris függvények [1]). *Egy $\mathcal{F} : 2^V \to \mathbb{R}$ V alaphalmaz részhalmazain értelmezett halmazfüggvényt szubmodulárisnak, vagy teljesen szubmodulárisnak nevezünk, ha $\forall X, Y \subseteq V$ halmazpárra teljesül a következő egyenlőtlenség:*

$$\mathcal{F}(X) + \mathcal{F}(Y) \geq \mathcal{F}(X \cap Y) + \mathcal{F}(X \cup Y)$$

A szubmodularitást gyakran másképpen definiálják [1]:

1.1.2. Állítás. *Egy $\mathcal{F} : 2^V \to \mathbb{R}$ V alaphalmaz részhalmazain értelmezett halmazfüggvény pontosan akkor szubmoduláris, ha $\forall B \subseteq A \subseteq V$ halmazokra és $x \in \overline{\mathrm{A}}$ esetén igaz, hogy*

$$\mathcal{F}(A \cup x) - \mathcal{F}(A) \leq \mathcal{F}(B \cup x) - \mathcal{F}(B)$$

Bizonyítás. $\Rightarrow$
Tegyük fel, hogy

$$\mathcal{F}(X) + \mathcal{F}(Y) \geq \mathcal{F}(X \cap Y) + \mathcal{F}(X \cup Y) \quad \forall X, Y \subseteq V$$

Ezt átrendezve kapjuk, hogy

$$\mathcal{F}(X) - \mathcal{F}(X \cap Y) \geq \mathcal{F}(X \cup Y) - \mathcal{F}(Y) \quad \forall X, Y \subseteq V$$

Legyen $B \subseteq A \subseteq V$ és $x \in \overline{\mathrm{A}}$. Legyen $X = B \cup x$ és $Y = A$. Ekkor az $X \cap Y = B$, az $X \cup Y = A \cup x$. Tehát valóban igaz, hogy

$$\mathcal{F}(B \cup x) - \mathcal{F}(B) \geq \mathcal{F}(A \cup x) - \mathcal{F}(A)$$

$\Leftarrow$
Tegyük fel, hogy

$$\mathcal{F}(A \cup x) - \mathcal{F}(A) \leq \mathcal{F}(B \cup x) - \mathcal{F}(B) \quad \forall B \subseteq A \subseteq V \quad x \in \overline{\mathrm{A}}$$

Egyszerű átalakítással kapjuk:

$$\mathcal{F}(A \cup x) + \mathcal{F}(B) \leq \mathcal{F}(B \cup x) + \mathcal{F}(A) \quad \forall B \subseteq A \subseteq V \quad x \in \overline{\mathrm{A}}$$

Legyen $X = B \cup x$ és $Y = A$. Ekkor az X és Y halmazok uniója az $A \cup x$, a metszete pedig B. Ezt felhasználva pont a bizonyítandó egyenlőtlenséget kapjuk:

$$\mathcal{F}(X \cup Y) - \mathcal{F}(X \cap Y) \leq \mathcal{F}(X) + \mathcal{F}(Y)$$

□

Az 1.1.2-es állítás szemlélteti, hogy egy adott elem hozzávétele egy bővülő halmazsorozathoz nemnövekvő haszonnal jár. Ezt, a szubmoduláris függvények által kielégített tulajdonságot gyakran emlegetik a csökkenő hasznok elveként.
Néhány példát szeretnék megemlíteni, melyek tükrözik a szubmodularitás lényegét.

1.1.3. Példa. *Adott egy város, ahol bűnmegelőzés céljából szeretnének térfigyelő kamerákat elhelyezni. A V alaphalmaz jelölje azon helyeket, ahová ezeket a kamerákat fel lehet szerelni, és ennek egy S részhalmazára jelölje $\mathcal{F}(S)$ az S-beli helyekre elhelyezett kamerák együttes hatékonyságát. Ekkor $\mathcal{F}$ szubmoduláris, mivel teljesül a fent említett csökkenő hasznok elve.*

1.1.4. Példa. *Legyen $G(V, E)$ tetszőleges irányítatlan gráf, $c : E \to \mathbb{R}_0^+$ élköltség, továbbá $X \subseteq V$ esetén jelölje $d(X)$ az X vágás által meghatározott éleken szereplő súlyok összegét. Ekkor a d függvény szubmoduláris, ugyanis teljesül a következő egyenlőség:*

$$d(X) + d(Y) = d(X \cap Y) + d(X \cup Y) + 2d(X, Y),$$

ahol $d(X, Y)$ jelöli az $X \setminus Y$ és $Y \setminus X$ közötti élek súlyainak összegét. Az egyenlőségből pedig következik az 1.1.1.-es definíció.
Ha G irányított, akkor a d, illetve a ρ függvények is szubmodulárisak.

A továbbiakban gyakran monoton növő szubmoduláris függvényekről lesz szó, ugyanis egyes maximalizáló algoritmusok használni fogják a monotonitást.

Megjegyzés: A véges-értékű, monoton növő, szubmoduláris függvényeket polimatroid függvényeknek is szokás nevezni.

1.2. Szupermodularitás, modularitás

1.2.1. Definíció (Szupermoduláris függvények [2]). *Egy $\mathcal{F} : 2^V \to \mathbb{R}$ V alaphalmaz részhalmazain értelmezett halmazfüggvényt szupermodulárisnak, vagy teljesen szupermodulárisnak nevezünk, ha $-\mathcal{F}$ szubmoduláris.*

1.2.2. Példa. *$G(V, E)$ tetszőleges irányítatlan gráf, $c : E \to \mathbb{R}_0^+$ élköltség, továbbá $X \subseteq V$ esetén jelölje $D(X)$ az X csúcsai között futó éleken szereplő súlyok összegét. Ekkor a D függvény szupermoduláris.*

1.2.3. Definíció (Moduláris függvények [2]). *Az olyan szubmoduláris $\mathcal{F}$ függvényeket, melyek az 1.1.1.-es definícióban szereplő egyenlőtlenséget egyenlőséggel teljesítik, moduláris függvényeknek nevezzük.*

Más szóval ha egy függvény szubmoduláris és szupermoduláris is, akkor moduláris. Az olyan moduláris függvények, melyekre igaz, hogy $\mathcal{F}(\emptyset) = 0$, felírhatóak olyan alakban, hogy

$$\mathcal{F}(X) = \sum_{x \in X} \mathcal{F}(x) \quad \forall X \subseteq V,$$

ami azt jelenti, hogy $\mathcal{F}$-et az egyelemű halmazokon vett értékei meghatározzák.

1.2.4. Állítás ([1]). *Egy $\mathcal{F}$ véges értékű szubmoduláris függvény, melyre teljesül, hogy az üres halmazon eltűnik ($\mathcal{F}(\emptyset) = 0$) pontosan akkor moduláris, ha*

$$\mathcal{F}(x) + \mathcal{F}(V \setminus x) = \mathcal{F}(V) \quad \forall x \in V$$

Bizonyítás. $\Rightarrow$ Tegyük fel, hogy $\mathcal{F}$ moduláris. Ekkor a fentiek szerint az egyelemű halmazokon vett értékei meghatározzák, ezért az $\mathcal{F}(V \setminus x)$ felírható a következő alakban: $\mathcal{F}(V \setminus x) = \sum_{y \in V \setminus x} \mathcal{F}(y)$. Ebből adódik, hogy $\mathcal{F}(x) + \mathcal{F}(V \setminus x) = \mathcal{F}(x) + \sum_{y \in V \setminus x} \mathcal{F}(y) = \sum_{y \in V} \mathcal{F}(y)$, ami pedig pont $\mathcal{F}(V)$-vel egyezik meg.
$\Leftarrow$ Indirekt tegyük fel, hogy $\mathcal{F}$ szubmoduláris, véges értékű, és teljesül rá az állításban megfogalmazott egyenlőség, de mégsem moduláris, azaz léteznek olyan A,

B halmazok, hogy $B \subseteq A \subseteq V$, de $\mathcal{F}(A \cup x) - \mathcal{F}(A) < \mathcal{F}(B \cup x) - \mathcal{F}(B)$. Viszont a szubmoduláris függvényekre vonatkozó 1.1.2.-es állítás, az 1.2.3.-as állítás feltétele és az indirekt feltevés alapján ellentmondásra jutunk: $\mathcal{F}(x) = \mathcal{F}(V) - \mathcal{F}(V \setminus x) \leq \mathcal{F}(A \cup x) - \mathcal{F}(A) < \mathcal{F}(B \cup x) - \mathcal{F}(B) \leq \mathcal{F}(\emptyset \cup x) - \mathcal{F}(\emptyset) = \mathcal{F}(x)$.

□

1.3. Fontos tulajdonságok

Az előzőekben definiált függvények néhány fontos tulajdonságát szeretném megemlíteni. [2]

1.3.1. Állítás. *Ha az $\mathcal{F}$ halmazfüggvény szubmoduláris, akkor az $\mathcal{F}(V \setminus X)$ is szubmoduláris, a $-\mathcal{F}(X)$ pedig szupermoduláris.*

1.3.2. Állítás. *Szubmoduláris függvények nemnegatív lineáris kombinációja is szubmoduláris, azaz ha $\mathcal{F}$, $\mathcal{G}$ szubmodulárisak és $\alpha, \beta \geq 0$, akkor a $\mathcal{H}(S) := \alpha\mathcal{F}(S) + \beta\mathcal{G}(S)$ függvény is szubmoduláris.*

Bizonyítás. Az 1.1.1.-es definícióból könnyű meggondolni. □

1.3.3. Lemma. *Ha $\mathcal{F}$ monoton növő szubmoduláris függvény, akkor $\forall X, Y \subseteq V$ esetén $\mathcal{F}(X) \leq \mathcal{F}(Y) + \sum_{k \in X \setminus Y} \rho_k(Y)$, ahol $\rho_k(Y) = \mathcal{F}(Y \cup k) - \mathcal{F}(Y)$.*

A következő konstrukcióval egy tetszőleges függvényt monoton függvénnyé alakíthatunk.

1.3.4. Állítás. *Legyen $\mathcal{F}$ a V alaphalmaz részhalmazain definiált tetszőleges halmazfüggvény. Legyen*

$$\mathcal{F}_{mon}(X) := \min_{Y \subseteq X} \mathcal{F}(Y)$$

Ekkor $\mathcal{F}_{mon}$ monoton csökkenő.

1.3.5. Állítás. *Ha $\mathcal{F}$ szubmoduláris, akkor $\mathcal{F}_{mon}$ is szubmoduláris.*

1.3.6. Állítás. *Legyenek az $\mathcal{F}$ és $\mathcal{G}$ szubmoduláris függvények olyanok, hogy $\mathcal{F} - \mathcal{G}$ monoton, akkor $\min\{\mathcal{F}, \mathcal{G}\}$ is szubmoduláris.*

A konkáv függvények és a szubmodularitás kapcsolatát mutatja az alábbi állítás, amit a továbbiakban többször is használni fogunk, amikor szövegtömörítéshez hasznos szubmoduláris függvényeket definiálunk. [3]

1.3.7. Tétel. *Legyen $\mathcal{F} : 2^V \to \mathbb{R}$ monoton növő szubmoduláris halmazfüggvény, $f : \mathbb{R} \to \mathbb{R}$ monoton növő konkáv függvény. Ekkor a két függvény kompozíciója, azaz $\mathcal{F}' = f \circ \mathcal{F} : 2^V \to \mathbb{R}$ monoton növő és szubmoduláris.*

Bizonyítás. Először azt látom be, hogy $\mathcal{F}'$ monoton növő: legyen $A \subseteq B \subseteq V$ tetszőleges. Ekkor $\mathcal{F}$ monotonitása miatt $\mathcal{F}(A) \leq \mathcal{F}(B) \Rightarrow f$ monotonitása miatt $f(\mathcal{F}(A)) \leq f(\mathcal{F}(B))$, ami pedig pont azt jelenti, hogy $\mathcal{F}'(A) \leq \mathcal{F}'(B)$. A szubmodularitás is látszik, ugyanis $\mathcal{F}$ szubmoduláris (teljesül rá az 1.1.1. definíció), monoton, ezért $\mathcal{F}(X \cap Y) \leq \mathcal{F}(X) \leq \mathcal{F}(Y) \leq \mathcal{F}(X \cup Y)$ és f konkávitása miatt $\frac{f(\mathcal{F}(X \cap Y)) + f(\mathcal{F}(X \cup Y))}{2} \leq \frac{f(\mathcal{F}(X)) + f(\mathcal{F}(Y))}{2}$, amiből már következik az $\mathcal{F}'$ szubmodularitása.

□

2. Monoton szubmoduláris függvények maximalizálása

Legyen adott egy V alaphalmaz, egy $\mathcal{F} : 2^V \to \mathbb{R}$ halmazfüggvény és egy $K \in \mathbb{R}$ szám. A feladat egy olyan $S \subseteq V$ részhalmaz megadása, amire ez az $\mathcal{F}$ függvény maximális és $|S| \leq K$.
Szubmoduláris függvények minimalizálása megoldható polinom időben [10], viszont a maximalizálás egy NP-nehéz feladat [3]. Ennek ellenére bizonyos esetekben mégis jól kezelhető a probléma:
Ha az $\mathcal{F}$ függvény monoton növő szubmoduláris, akkor is NP-nehéz a feladat (2.0.2. Tétel), de erre van egy bizonyítottan $1 - \frac{1}{e} \sim 0,63$-approximáló mohó algoritmus (Fischer, Nemhauser, Wolsey 1978). Ezen algoritmus [13] lépései:

Legyen $S = \emptyset$
while $|S| \leq K$ **do**
  Adjuk hozzá S-hez azt az i elemet, ami maximalizálja $\mathcal{F}(S \cup i)$-t
end while

2.0.1. Tétel ([13]). *Ha $\mathcal{F}$ monoton növő, szubmoduláris és $\mathcal{F}(\emptyset) = 0$, akkor a fenti mohó algoritmus olyan S megoldást ad, amire*

$$\mathcal{F}(S) \geq \left(1 - \frac{1}{e}\right) \cdot OPT$$

ahol $OPT = \max_{S:|S|\leq K} \mathcal{F}(S)$.

Bizonyítás. Jelölje az algoritmus i. iterációja utáni halmazt S_i, az optimális halmaz pedig legyen $S^* = \{v_1, v_2, ..., v_k\}$. A monotonitást és a szubmoduláris függvényekre vonatkozó 1.1.2-es állítást felhasználva kapjuk, hogy

$$\mathcal{F}(S^*) \leq \mathcal{F}(S_i \cup S^*) = \mathcal{F}(S_i \cup \{x_1, x_2, ..., x_k\}) \leq$$

$$\leq \mathcal{F}(S_i) + (\mathcal{F}(S_i \cup x_1) - \mathcal{F}(S_i)) + (\mathcal{F}(S_i \cup \{x_1, x_2\}) - \mathcal{F}(S_i \cup x_1)) + ... \leq$$

$$\leq \mathcal{F}(S_i) + (\mathcal{F}(S_i \cup x_1) - \mathcal{F}(S_i)) + (\mathcal{F}(S_i \cup x_2) - \mathcal{F}(S_i)) + ... \leq$$

$$\leq \mathcal{F}(S_i) + K \cdot (\mathcal{F}(S_{i+1}) - \mathcal{F}(S_i))$$

Ezt rendezzük át a következőképpen:

$$\mathcal{F}(S_{i+1}) - \mathcal{F}(S_i) \geq \frac{1}{K}(\mathcal{F}(S^*) - \mathcal{F}(S_i))$$

A bizonyítás következő lépésében indukcióval belátjuk, hogy

$$\mathcal{F}(S_i) \geq (1 - (1 - \frac{1}{K})^i) \cdot \mathcal{F}(S^*)$$

Ez $i = 0$-ra triviális. Tegyük fel, hogy i-re igaz, ekkor azt szeretnénk látni, hogy $i + 1$-re is igaz.

$$\mathcal{F}(S_{i+1}) \geq \mathcal{F}(S_i) + \frac{1}{K}(\mathcal{F}(S^*) - \mathcal{F}(S_i)) = (1 - \frac{1}{K}) \cdot \mathcal{F}(S_i) + \frac{1}{K} \cdot \mathcal{F}(S^*) \geq$$

$$\geq (1 - \frac{1}{K}) \cdot (1 - (1 - \frac{1}{K})^i) \cdot \mathcal{F}(S^*) + \frac{1}{K} \cdot \mathcal{F}(S^*) = (1 - (1 - \frac{1}{K})^{i+1}) \cdot \mathcal{F}(S^*)$$

Így az indukcióval készen vagyunk, és abból, amit beláttunk következik, hogy

$$\mathcal{F}(S) \geq (1 - (1 - \frac{1}{K})^K) \cdot \mathcal{F}(S^*) \geq (1 - \frac{1}{e}) \cdot \mathcal{F}(S^*) = (1 - \frac{1}{e}) \cdot OPT$$

□

2.0.2. Tétel (Feige, 1998 [13]). *A fönti feladatra* $(1 - \frac{1}{e}) + \varepsilon$ *(*$\varepsilon > 0$ *tetszőleges) közelítő megoldást találni NP-nehéz.*

3. Szubmodularitás a szövegfeldolgozásban

3.1. Tömörítés adott felső korláttal

Legyen adott egy tömöríteni kívánt szöveg és egy V alaphalmaz, ami a szöveg összes mondatát (vagy más nyelvi egységét) tartalmazza. Elképzelhető olyan feladat is, hogy több dokumentumot kapunk, és ezek egészén kell elvégezni a lényegkiemelési eljárást. Ekkor az alaphalmaz elemei lehetnek maguk a dokumentumok is (*"multi-document summarization"*).
A feladat egy olyan $S \subseteq V$ részhalmaz kiválasztása, ami az egész V-t reprezentálja. Természetesen azt szeretnénk, hogy a kiválasztott részhalmaz számossága minél kisebb legyen. Sok feladatban ezért egy felső korláttal szorítják meg S elemszámát. Ezt a következőképpen lehet megtenni:

$$\sum_{i \in S} c_i \leq b,$$

ahol $c_i \in \mathbb{R}_0^+$ jelöli az i. egység beválasztásával járó költséget (például a szavak vagy karakterek száma az adott mondatban vagy más szövegegységben), a $b \in \mathbb{R}^+$ pedig az úgynevezett "teherbírásunkat" (pl. a tömörítés által tartalmazható szavak vagy karakterek maximális száma). Legyen egy $\mathcal{F} : 2^V \to \mathbb{R}$ halmazfüggvény úgy, hogy $\mathcal{F}(S)$ az S szövegtömörítés minőségét méri. Ekkor formalizálva azt mondhatjuk, hogy a feladat

3.1.1. Feladat. *Olyan $S^* \subseteq V$ részhalmaz megtalálása, amire:*

$$\mathcal{F}(S^*) = \max_{S \subseteq V} \mathcal{F}(S) \quad \textit{melyekre} \quad \sum_{i \in S} c_i \leq b$$

Ha a c_i értékek mindegyike 1, akkor a 2. fejezetben ismertetett feladatot kapjuk, és mivel az NP-nehéz, ezért a 3.1.1.-es általánosabb feladat is NP-nehéz. Tehát a 2. fejezetben bemutatott approximációs algoritmus csupán abban az esetben használható, ha minden $c_i = 1$ és az $\mathcal{F}$ monoton növő szubmoduláris függvény. A továbbiakban azt részletezem, hogy az általánosabb esetben milyen módszereket, modelleket célszerű használni.

3.1.1. Maximális hozzáadott relevancia (MMR)

A szövegtömörítési feladatban gyakran megjelennek a mohó algoritmusok. Egy nagyon fontos és széleskörűen használt eljárás az MMR (Maximal Marginal Relevance), melynek az alapja az, hogy az alaphalmaz elemeit hasznosságuk szerint (alkalmazkodva a felhasználói elváráshoz) csökkenő sorrendbe rendezze. Egy kis átalakítással könnyen használható ez az algoritmus egy lényegkiemelési feladat megoldására, melynek során a mohó algoritmus kiválasztja a legrelevánsabb mondatot (vagy más egységet) és közben elkerüli a felesleges ismétlődéseket úgy, hogy nem választ olyan mondatokat, amik túl hasonlóak a már kiválasztottakhoz. Ehhez a Carbonell és Goldstein (1998) által definiált algoritmus minden lépésben azt, a tömörítés által még nem tartalmazott elemet, mondatot ($\hat{s}$) választja be, melyre

$$\hat{s} = arg \max_{s \in V \setminus S} \{\lambda \cdot Sim_1(s, Q) - (1 - \lambda) \max_{s_j \in S} Sim_2(s, s_j)\},$$

ahol V az alaphalmaz, S jelöli a már kiválasztott mondatok halamzát, Q a tömörítéssel kapcsolatos elvárás (ez gyakran felhasználófüggő, érdemes úgy gondolni rá, mint a várt tömörítés kulcsszavait tartalmazó szöveg), Sim_1 és Sim_2 két hasonlósági függvény, amik lehetnek ugyanazok, vagy különböző metrikák is (a gyakorlatban általában a koszinusz hasonlóságot használják). A $Sim_1(s, Q)$ az s egység és a Q elvárás közötti hasonlóságot, a $\max_{s_j \in S} Sim_2(s, s_j)$ pedig az s maximális távolságát méri a már kiválasztott elemektől. Utóbbi az ismétlődés csökkentésére szolgál. A definícióban szereplő λ tetszőleges $[0, 1]$ intervallumbeli együttható.
Látszik, hogy az MMR eljárás során annak, hogy milyen értéket adtunk a λ-nak, fontos szerepe van, függ a tömörítési céltól. Ezalatt azt értem, hogy ha a $\lambda = 1$ választással éltünk, akkor az algoritmus a mondatokat csupán Q-relevancia szerint sorolja csökkenő sorrendbe, a $\lambda = 0$ esetben pedig csak az egymástól vett különbözőséget nézi, tehát ha az a fontosabb szempont, hogy a tömörítés a Q elváráshoz hasonlítson, akkor inkább egy 1-hez közelebbi értéket célszerű megadni a λ-nak, ha pedig azt tartjuk lényegesebbnek, hogy ne legyen ismétlődés a tömörítésben, akkor egy félnél kisebb, 0 közeli értékadás ajánlott. A gyakorlatban leggyakrabban használatos λ értékek: $0,3$; $0,5$; $0,7$.

Az MMR eljárás lépései során maximalizált kifejezés segítségével a következő $\mathcal{F} : 2^V \to \mathbb{R}$ halmazfüggvényt definiálhatjuk:

$$\mathcal{F}_{mmr}(S) = \lambda \sum_{s_i \in S} Sim_1(s_i, Q) - (1 - \lambda) \sum_{s_i \in S} \max_{\substack{j \in S; \\ i \neq j}} Sim_2(s_i, s_j)$$

3.1.2. Tétel ([3]). *$\mathcal{F}_{mmr}$ szubmoduláris.*

Bizonyítás. A bizonyításhoz elég meggondolni, hogy a $\sum_{s_i \in S} Sim_1(s_i, Q)$ és a $-\sum_{s_i \in S} \max_{\substack{j \in S; \\ i \neq j}} Sim_2(s_i, s_j)$ két szubmoduláris függvény. Az 1.3.2.-es tétel miatt a nemnegatív lineáris kombinációjuk is szubmoduláris. □

Viszont fontos megjegyezni, hogy ebben az esteben az $\mathcal{F}_{mmr}$ függvény nem monoton növő. A következő fejezetben ennek a függvénynek egy speciális esetével ismerkedünk meg.

Megjegyzés: A kérdések mesterséges megválaszolása során az egyik fontos lépés a dokumentumokból a válaszhoz szükséges információ kigyűjtése, melyre gyakran az MMR eljárást használják. Ekkor a Q maga a kérdés, a Sim_1 definiálásához Hanning ablakfüggvényt használnak, a Sim_2 pedig a már említett koszinusz hasonlóság. [12]

3.1.2. Módosított mohó algoritmus

Lin és Bilmes [5] egy olyan $G(V, E)$ irányítatlan, súlyozott gráffal szemléltették a tömöríteni kívánt szöveget, melynek a csúcsai az egyes mondatoknak felelnek meg, az $ij \in E$ élen szereplő nemnegatív súly ($w_{i,j}$) pedig az i és j mondatok közötti hasonlóságot jelzi. Legyen $\mathcal{F}_{cut}$ a gráf csúcshalmazán értelmezett, úgynevezett *"graph-cut"* függvény a következőképpen értelmezve:

$$\mathcal{F}_{cut}(S) = \sum_{i \in S} \sum_{j \in V \setminus S} w_{i,j} \quad \forall S \subseteq V$$

Ez a függvény, ami lényegében az adott S tömörítés és a többi mondat közötti hasonlóságot méri, kielégíti a szubmodularitás definícióját az 1.1.4.-es példa szerint ($d = \mathcal{F}_{cut}$). Az előző bekezdésben definiált MMR eljárást alapul véve egy kis

módosítást eszközöltek az $\mathcal{F}_{cut}$ célfüggvényen:

$$\mathcal{F}^*_{cut}(S) = \sum_{i \in S} \sum_{j \in V \setminus S} w_{i,j} - \lambda \cdot \sum_{i,j \in S; i \neq j} w_{i,j} \quad \lambda \geq 0 \quad \forall S \subseteq V$$

Mivel az $\mathcal{F}_{cut}$ függvény szubmoduláris és a $\mathcal{G} = -\lambda \cdot \sum_{i,j \in S; i \neq j} w_{i,j}$ függvény is szubmoduláris (ugyanis $-\mathcal{G}$ szupermoduláris az 1.2.2.-es példa alapján), ezért az 1.3.2.-es állítás miatt $\mathcal{F}^*_{cut}$ szubmoduláris. Ezen célfüggvényt használva a következő "módosított" mohó algoritmust [5] lehet adni a 3.1.1.-es feladatra:

Legyen $G = \emptyset$ és $U = V$
while $U \neq 0$ **do**
  $k := argmax_{l \in U} \frac{\mathcal{F}(G \cup l) - \mathcal{F}(G)}{c_l^r}$
  if $c_k + \sum_{i \in G} c_i \leq b$ **és** $\mathcal{F}(G \cup k) - \mathcal{F}(G) \geq 0$ **then**
    $G := G \cup k$
    $U := U \setminus k$
  else
    $U := U \setminus k$
  end if
end while
$v^* := argmax_{v \in V c_v \leq b} \mathcal{F}(v)$
return $G_{\mathcal{F}} = argmax\{\mathcal{F}(v^*); \mathcal{F}(G)\}$

Az algoritmus approximációjáról szóló tétel bizonyításához elengedhetetlen az alábbi lemma, és az ebből következő tétel ismertetése, ehhez használjuk a következő jelöléseket: legyen S^* a 3.1.1.-es feladat optimális megoldása, $G_{\mathcal{F}}$, G az algoritmus leírása során definiált halmazok, k_i az i-edik iterációban G-hez adott elem és G_i az így kapott halmaz ($i = 1, ..., |G|$). Továbbá legyen $\mathcal{F}$ monoton növő szubmoduláris halmazfüggvény és használjuk az 1.3.3.-as lemmában említett ρ_k jelölést, azaz $\rho_k(S) = \mathcal{F}(S \cup k)$.

3.1.3. Lemma. *Ha $\mathcal{F}$ monoton növő szubmoduláris, akkor $0 \leq r \leq 1$ esetén*

$$\mathcal{F}(S^*) - \mathcal{F}(G_{i-1}) \leq \frac{b^r |S^*|^{1-r}}{c_{k_i}^r} (\mathcal{F}(G_i) - \mathcal{F}(G_{i-1})) \qquad \forall i = 1, ..., |G|$$

$r \geq 1$ esetén

$$\mathcal{F}(S^*) - \mathcal{F}(G_{i-1}) \leq \Big(\frac{b}{c_{k_i}}\Big)^r (\mathcal{F}(G_i) - \mathcal{F}(G_{i-1})) \qquad \forall i = 1, ..., |G|$$

Bizonyítás. Az algoritmus harmadik sora alapján

$$\frac{\rho_u(G_{i-1})}{c_u^r} \leq \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} \qquad \forall u \in S^* \setminus G_{i-1}$$

Ezért a $0 \leq r \leq 1$ esetben a következő egyenlőtlenséget kapjuk:

$$\sum_{u \in S^* \setminus G_{i-1}} \rho_u(G_{i-1}) \leq \sum_{u \in S^* \setminus G_{i-1}} \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} c_u^r = \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} \sum_{u \in S^* \setminus G_{i-1}} c_u^r$$

Felhasználva, hogy a $g(x) = x^r$ függvény $x > 0$ és $0 \leq r \leq 1$ esetén konkáv, a következő felső becslést kapjuk az eddigiekre:

$$\frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} \sum_{u \in S^* \setminus G_{i-1}} c_u^r \leq \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} |S^* \setminus G_{i-1}| \Big(\frac{\sum_{u \in S^* \setminus G_{i-1}} c_u}{|S^* \setminus G_{i-1}|} \Big)^r \leq$$

$$\leq \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} |S^* \setminus G_{i-1}|^{1-r} \Big(\sum_{u \in S^* \setminus G_{i-1}} c_u \Big)^r \leq \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} |S^*|^{1-r} \Big(\sum_{u \in S^*} c_u \Big)^r \leq$$

$$\leq \frac{\rho_{k_i}(G_{i-1})}{c_{k_i}^r} |S^*|^{1-r} b^r$$

Ahol felhasználtuk, hogy az S^* a feladat optimális megoldása, ezért az elemeinek súlyainak az összege nem haladja meg az adott felső korlátot. Ha az 1.3.3.-as lemmát $X = S^*$ és $Y = G_{i-1}$ választással alkalmazzuk és a $\rho_{k_i}(G_{i-1})$ definícióját használjuk, akkor átrendezés után a bizonyítani kívánt egyenlőtlenséget kapjuk. Az $r \geq 1$ esetben hasonlóan járhatunk el.

□

3.1.4. Tétel. *Ha $\mathcal{F}$ monoton növő szubmoduláris, akkor $0 \leq r \leq 1$ esetén*

$$\mathcal{F}(G_{i-1}) \geq \Big(1 - \prod_{j=1}^{i} \big(1 - \frac{c_{k_j}^r}{b^r |V|^{1-r}}\big)\Big)\mathcal{F}(S^*) \qquad \forall i = 1, ..., |G|$$

$r \geq 1$ esetén

$$\mathcal{F}(G_{i-1}) \geq \Big(1 - \prod_{j=1}^{i} \big(1 - \big(\frac{c_{k_j}}{b}\big)^r\big)\Big)\mathcal{F}(S^*) \qquad \forall i = 1, ..., |G|$$

Bizonyítás. A tétel teljes indukcióval könnyen bizonyítható. Az $i = 1$ eset az előző lemma (3.1.3), a $G_0 = \emptyset$ és a $|V| \geq |S^*|$ felhasználásával nyilvánvalóan teljesül, továbbá az $(i-1)$-ről $i-re$ való öröklődéshez is csupán az fenti 3.1.3.-as lemmára van szükség, és persze az indukciós feltevésre. □

3.1.5. Tétel ([5])**.** *Ha $\mathcal{F}$ monoton növő szubmoduláris függvény, akkor a fenti algoritmus r=1 esetén $(1 - \frac{1}{\sqrt{e}}) \sim 0.39$ approximácós, azaz*

$$\mathcal{F}(G_{\mathcal{F}}) \geq (1 - \frac{1}{\sqrt{e}})\mathcal{F}(S^*)$$

Bizonyítás. A tétel bizonyítása esetszétválasztással történik:
1. eset: $\exists v \in V : \mathcal{F}(v) > \frac{1}{2}\mathcal{F}(S^*)$. Ekkor az algoritmus utolsó sorának értelmében: $\mathcal{F}(G_{\mathcal{F}}) \geq \mathcal{F}(v) > \frac{1}{2}\mathcal{F}(S^*)$.
2. eset: $\forall v \in V : \mathcal{F}(v) \leq \frac{1}{2}\mathcal{F}(S^*)$ Ekkor ismét két esetet különböztetünk meg:
2.a eset: $\sum_{v \in G} c_v \leq \frac{1}{2}b$. Ekkor $\forall v \notin G$ súlya $> \frac{1}{2}b$, mert ha ez nem lenne igaz, akkor lenne olyan v, melyet G-hez hozzávéve a maximalizálandó érték nőne és még mindig nem sérülne meg a korlátossági feltétel. Ebből következik, hogy az $S^* \setminus G$ legfeljebb egyelemű halmaz, ugyanis ellenkező esetben a $\sum_{v \in S^*} c_v > b$ egyenlőtlenség adódna. Tehát a 2. esetben megfogalmazott feltétel miatt: $\mathcal{F}(S^* \setminus G) \leq \frac{1}{2}\mathcal{F}(S^*)$. Az $\mathcal{F}$ függvény szubmodularitását használva: $\mathcal{F}(S^* \setminus G) + \mathcal{F}(S^* \cap G) \geq \mathcal{F}(S^*)$ Ebből adódik, hogy $\mathcal{F}(S^* \cap G) \geq \frac{1}{2}\mathcal{F}(S^*)$. Az $\mathcal{F}$ függvény monotonitása miatt: $\mathcal{F}(G_{\mathcal{F}}) \geq \mathcal{F}(G) \geq \mathcal{F}(S^* \cap G) \geq \frac{1}{2}\mathcal{F}(S^*)$
2.b eset: $\sum_{v \in G} c_v > \frac{1}{2}b$. Ekkor használjuk az 3.1.4.-es tételt $0 \leq r \leq 1$ és $i = |G|$

választással:

$$\mathcal{F}(G) \geq \Big(1 - \prod_{j=1}^{|G|}(1 - \frac{c_{k_j}^r}{b^r |V|^{1-r}})\Big)\mathcal{F}(S^*) \geq \Big(1 - \prod_{j=1}^{|G|}(1 - \frac{c_{k_j}^r |V|^{r-1}}{2^r (\sum_{j=1}^{|G|} c_{k_j})^r})\Big)\mathcal{F}(S^*) \geq$$

$$\geq \Big(1 - (1 - \frac{|V|^{r-1}}{2^r |G|^r})^{|G|}\Big)\mathcal{F}(S^*) \geq \Big(1 - e^{-\frac{1}{2}\left(\frac{|V|}{2|G|}\right)^{r-1}}\Big)\mathcal{F}(S^*)$$

A harmadik egyenlőtlenséghez a Lagrange-féle multiplikátor módszer szükséges, az utolsónál pedig az $e^{-x} \geq 1 - x$ egyenlőtlenséget alkalmazzuk. Így $r = 1$ esetén $\mathcal{F}(G_{\mathcal{F}}) \geq (1 - \frac{1}{\sqrt{e}})\mathcal{F}(S^*)$. Ezt szerettük volna belátni.

□

Több kérdés is felmerül az algoritmussal kapcsolatban. Az egyik az, hogy miért ezt az algoritmust célszerű használni és miért nem a 2. fejezetben bemutatottat annak ellenére, hogy az egy jobb approximációval bír. Erre a válasz az, hogy a második fejezetben tárgyalt algoritmus approximációs faktora akkor igaz, ha az egyes egységekhez tartozó súlyok megegyeznek, viszont a tömörítési feladatnál ez nem feltétlenül igaz. Habár igaz, hogy különböző súlyok esetén is megadható lenne egy $(1 - \frac{1}{e})$ approximáló algoritmus [11], de ennek futásideje $O(|V|^5)$, ezért a gyakorlatban inkább az $(1 - \frac{1}{\sqrt{e}})$-s algoritmust alkalmazzák.
A másik jogos kérdés az, hogy mi a szerepe az $r \geq 0$ paraméternek, és az algoritmus utolsó két sorának, azaz a v^* és $G_{\mathcal{F}}$ definiálásának. Válaszként tekintsük ugyanezt az eljárást, az utolsó két sora és r nélkül és vegyük a következő példát: $V = \{x, y\}$, $\mathcal{F}(x) = 1$, $c_x = 1$, $\mathcal{F}(y) = p$, $c_y = p + 1$, $b = p + 1$. Ekkor az algoritmus outputja a, 1 célfüggvényértékkel, viszont az optimális célfüggvényérték p. Tehát az algoritmus approximációs hányadosa nem véges. Azzal, hogy az algoritmus megkeresi azt a $v^* \in V$ elemet, melyre $\mathcal{F}(v^*)$ maximális úgy, hogy $c_{v^*} \leq b$, és ha ez a maximumérték nagyobb, mint az $\mathcal{F}(G)$, akkor ezt adja outputként, már egy közelítő algoritmust kapunk (ez az előző tétel), r jó megválasztásával pedig még pontosabb közelítések is elérhetőek.
Fontos kihangsúlyozni, hogy a 3.1.5.-ös tételben megfogalmazott approximáció csupán akkor igaz, ha az $\mathcal{F}$ függvény szubmoduláris és monoton növő. Ez nem

feltétlenül teljesül az $\mathcal{F}^*_{cut}$ függvényre, viszont az $\mathcal{F}^*_{cut}$ gráfra építve definiálhatóságának következménye az, hogy ha $|S|$ "kicsi", akkor nagy valószínűséggel optimum közeli megoldást kapunk (teljesül a 3.1.5.-ös tétel):

3.1.6. Tétel ([5]). *Tegyük fel, hogy az élek súlyai korlátosak, azaz $w_{i,j} \in [0,1]$ teljesül $\forall ij \in E$ esetén és független, azonos eloszlásúak $\mu = \mathbb{E}(w_{ij})$ várható értékkel. Ekkor az $\mathcal{F}^*_{cut}$ függvénnyel futtatva az algoritmust a 3.1.5.-ös tétel legalább P valószínűséggel teljesül, ahol*

$$P = 1 - exp\{-\frac{2\big(|V| - (\alpha+1)\beta\big)^2\mu^2}{|V| + (\alpha^2-1)\beta} + lnb\},$$

$\alpha = 2\lambda + 1$, $\beta = 2b - 1$.

3.1.3. Koncepció alapú tömörítés

Az úgynevezett koncepció alapú ("concept-based") tömörítés során is szubmoduláris függvény jelenik meg [3]. Egyre jobban látszódik, hogy igaz a bevezetésben írt mondat arról, hogy a szubmoduláris függvények természetesnek mondhatóak a tömörítéses feladatok során.
Legyen a Γ "koncepciók" egy halmaza, az S pedig mondatok egy részhalmaza és jelölje $\Gamma(S)$ az S által tartalmazott koncepciók halmazát. Legyen $d : \Gamma \to \mathbb{R}^+$ függvény olyan, hogy d_i az i. koncepció fontosságát jelöli. Ekkor az S részhalmazt értékelő $\mathcal{F}_{concept} : 2^V \to \mathbb{R}$ halmazfüggvény legyen a következő:

$$\mathcal{F}_{concept}(S) := \sum_{i \in \Gamma(S)} d_i$$

Ez az $\mathcal{F}_{concept}$ függvény szubmoduláris, ugyanis ha egy egyre bővülő halmazsorozathoz vesszük hozzá ugyanazt az elemet, akkor az nemnövekvő haszonnal jár, vagyis teljesül az 1.1.2.-es állítás. Mivel a súlyok nemnegatív értékeket vehetnek fel, ezért a monoton növő tulajdonság is teljesül. A koncepció-alapú tömörítés során ezt az $\mathcal{F}$ függvényt kell maximalizálni megengedett halmazokra, azaz olyan S halmazokra, melyekre $\sum_{i \in S} c_i \leq b$.

3.2. Tömörítés adott alsó korláttal

Egy jó tömörítésnél nemcsak az a fontos, hogy rövid legyen, hanem elengedhetetlen az is, hogy az összes, a szövegben található lényeges információt tartalmazza. Ezt formalizálva úgy mondhatjuk, hogy:

3.2.1. Feladat. *Olyan S^* részhalmazát keressük a V alaphalmaznak, melyre:*

$$\mathcal{F}(S^*) = \min_{S \subseteq V} \sum_{i \in S} c_i \quad melyekre \quad \mathcal{F}(S) = \alpha,$$

ahol a c_i az i. egység (mondat) költsége, az $\mathcal{F}$ halmazfüggvény pedig az alaphalmaz részhalmazaihoz az adott részhalmaz által tartalmazott információ mennyiségét rendeli hozzá. Az $\alpha = \mathcal{F}(V)$, azaz a szöveg által tartalmazott információk mennyisége. (Előfordulhat az is, hogy csak azt követeli meg a feladat, hogy az információknak legalább egy adott része legyen benne a kiválasztott részhalmazban, ekkor egyenlőség helyett egyenlőtlenséggel írjuk fel a fenti feladatot.)

3.2.1. Wolsey algoritmusa

Tegyük fel, hogy az $\mathcal{F}$ függvény monoton növő szubmoduláris. A 3.2.1.-es feladathoz megadható egy ekvivalens egész értékű lineáris program [6]:

$$Z_I = min \sum_{i \in V} c_i \cdot y_i$$

$$\sum_{i \in V} \rho_i(S) y_i \geq \alpha - \mathcal{F}(S) \quad \forall S \subseteq V \quad y_i \in \{0,1\} \quad i \in V,$$

ahol $\rho_i(S) = \mathcal{F}(S \cup i) - \mathcal{F}(S)$ az 1.3.3.-as lemmából ismert jelölés.
A két feladat ekvivalenciájáról szól a következő tétel [6].

3.2.2. Tétel. *Egy $X \subseteq V$ halmaz pontosan akkor megengedett megoldása a 3.2.1.-es feladatnak, ha a karakterisztikus vektora (y^X) megengedett megoldása a Z_I lineáris programnak.*

Bizonyítás. $\Rightarrow$ Tegyük fel, hogy az X halmazra $\mathcal{F}(X) = \alpha$. Ekkor

$$\sum_{i \in V} \rho_i(S) y_i^X = \sum_{i \in X \setminus S} \rho_i(S) \geq \mathcal{F}(X) - \mathcal{F}(S) = \alpha - \mathcal{F}(S)$$

Az első egyenlőség azért igaz, mivel $y_i^X = 0$ ha $i \notin X$, különben 1, és $\rho_i(S) = 0$ ha $i \in S$. Az egyenlőtlenség az 1.3.3.-as lemma következménye, a második egyenlőség pedig a feltevésből adódik.
$\Leftarrow$ Tegyük fel, hogy az y^X karakterisztikus vektor megengedett megoldása a Z_I lineáris programnak, ekkor az előző irányhoz hasonlóan meggondolható, hogy

$$0 = \sum_{i \in V} \rho_i(X) y_i^X \geq \alpha - \mathcal{F}(X),$$

ahonnan $\mathcal{F}(X) = \alpha$ adódik.

□

Az egészértékű programozási feladat segítségével a következő approximáló mohó algoritmust [6] lehet adni a feladatra:

Legyen $S = \emptyset$
while $\mathcal{F}(S) < \alpha$ **do**
  $i := arg\min_{V \setminus S}\{\frac{c_i}{\rho_i(S)}\}$
  $S := S \cup \{i\}$
end while

3.2.3. Tétel ([6])**.** *Ha S^* jelöli az optimális megoldást, S pedig az algoritmus által adott megengedett megoldást, $i \in S$ jelöli azt az elemet, melyet az algoritmus utoljára adott hozzá a részhalmazhoz, akkor a következő felső becslés adható az algoritmus approximációjára:*

$$\mathcal{F}(S) \leq \big(1 + ln\{\frac{\alpha - \mathcal{F}(\emptyset)}{\alpha - \mathcal{F}(S - i)}\}\big)\mathcal{F}(S^*)$$

3.2.2. Minimális domináns halmaz keresése

A feladat egy másik megközelítése szerint [7] keressünk egy minimális domináns halmazt, ahol domináns halmaz alatt azt értjük, hogy az egész szöveg bármely mondatára igaz, hogy vagy benne van ebben a domináns halmazban, vagy legalább az egyik mondatához nagyon hasonlít. Ez alapján definiáljuk a $\delta : 2^V \to \mathbb{R}$ metrikát úgy, hogy $\delta(S)$ az összes olyan mondatot tartalmazza, ami vagy S-beli, vagy nagyon közel van S egy eleméhez. Ekkor azt mondjuk, hogy S domináns halmaz, ha $|\delta(S)| = |V|$. Ha az $\mathcal{F}$ halmazfüggvényt a következőképpen definiáljuk: $\mathcal{F}(S)_{dom} := |\delta(S)|$, akkor $\mathcal{F}_{dom}$ monoton növő és szubmoduláris. A minimális elemszámú domináns halmaz keresése feladat a 3.2.1.-es problémának egy speciális esete.

Ez a feladat is NP-nehéz, de adható rá egy approximációs mohó algoritmus, mely lényegében ugyanaz, mint az előző: egy üres halmazból indul ki és ha az i-edik iterációban az S még nem egy domináns halmaz, akkor hozzáveszi azt a v_i mondatot, ami még nincs S-ben és azon nem S-beli mondatok száma, melyek hasonítanak v_i-re maximális (Johnson, 1973). Ekkor az algoritmus [7] a következő approximációval bír:

3.2.4. Tétel ([7]). *Jelölje S az algoritmus által adott domináns halmazt, S^* pedig a minimális domináns halmazt. Ekkor $|S| \leq (1+ln|V|)|S^*|$, ahol V az alaphalmaz.*

3.3. Automatikus értékelés

A tömörítések automatikus értékelése rendkívül fontos, annak érdekében, hogy el lehessen kerülni a sok munkát igénylő és sokszor nem megfelelő kézi értékelést. A ROUGE (Recall-Oriented Understudy for Gisting Evaluation) gyakran használt a szövegtömörítések értékelésekor (Lin, 2014). Automatikusan meghatározza egy tömörítés minőségét úgy, hogy emberek által készített ideális tömörítésekhez hasonlítja. Különböző ROUGE mértékek vannak, például: ROUGE-L, ROUGE-W, ROUGE-S, ROUGE-N. Most az utóbbival fogok foglalkozni.

A ROUGE-N a hasonlóságot adja meg egy adott tömörítés (tömörítésjelölt) és az ideális/ajánlott tömörítések halmaza között. Ez formálisan úgy hangzik, hogy legyen V az alaphalmaz és legyen $S \subseteq V$ az alaphalmaz egy részhalmaza (ez lesz az a tömörítés, amit a ROUGE-N segítségével értékelni fogunk). Használjunk egy $c_e : 2^V \to \mathbb{Z}^+$ függvényt úgy, hogy $c_e(S)$ azt jelöli, hogy az e n-gram hányszor jelenik meg az S tömörítésben, ahol n-gram alatt n hosszú egységeket értünk, amik lehetnek: fonémák, szótagok, betűk, szavak. Tegyük fel, hogy adott K darab ideális tömörítés. Legyen az i. ideális/ajánlott tömörítés által tartalmazott n-gramok halmaza A_i $(i = 1, ..., K)$ és jelöljük $a_{e,i}$-vel azt, hogy az e n-gram hányszor szerepel az i. ajánlott tömörítésben. Ekkor:

3.3.1. Definíció.

$$\mathcal{F}_{ROUGE-N}(S) = \frac{\sum_{i=1}^{K} \sum_{e \in A_i} \min(c_e(S), a_{e,i})}{\sum_{i=1}^{K} \sum_{e \in A_i} a_{e,i}}$$

3.3.2. Tétel ([3])**.** *Az $\mathcal{F}_{ROUGE-N}$ függvény monoton növő szubmoduláris.*

Bizonyítás. Mivel $c_e(S)$ monoton növő moduláris és a $min(x, a)$ egy monoton növő konkáv függvénye x-nek, ezért az 1.3.7.-es állítást használva $\min(c_e(S), a_{e,i})$ monoton növő szubmoduláris függvény. Mivel a szummázás és a pozitív konstanssal való osztás nem befolyásolja a szubmodularitást (1.3.2.) és a monoton növő tulajdonságot sem, ezért $\mathcal{F}_{ROUGE-N}$ valóban monoton növő szubmoduláris. □

3.4. További monoton szubmoduláris célfüggvények

Egy tömörítésre akkor lehet azt mondani, hogy jó, ha visszaadja az eredeti szöveg lényegét és nem tartalmaz ismétléseket. Ennek szellemében a következőképpen modellezhető a tömörítés minősége:

$$\mathcal{F}(S) = \mathcal{L}(S) + \lambda\mathcal{R}(S),$$

ahol az $\mathcal{L}$ a V részhalmazain értelmezett halmazfüggvény azt mutatja meg, hogy az adott részhalmaz / tömörítés mennyire tükrözi az egész szöveg tartalmát (az információ mennyiségét méri), az $\mathcal{R}$ halmazfüggvény pedig az adott tömörítés sokszínűségét értékeli pozitívan (a változatosságot méri). Továbbá $\lambda \geq 0$ szabadon választott együtthatót jelöl. A továbbiakban ezen függvények lehetséges definíciói következnek, melyekről megmutatom, hogy monoton növő szubmodulárisak.[3]

3.4.1. Az információ mennyiségét mérő függvény

A már említett $\mathcal{L}(S)$ jelöli tehát azt, hogy az adott S tömörítés mennyire hasonlít a tömörítendő szöveghez, például a szöveg lényeges pontjai közül mennyit tartalmaz. Az rögtön látszik, hogy ez egy monoton növő függvény, mivel minél több mondatot tartalmazó tömörítést véve a szöveg hű visszaadása javul. A szubmodularitás is kézenfekvő, mivel teljesül az 1.1.2. állítás, ugyanis ha egy mondatot hozzáveszünk két tömörítéshez (úgy, hogy az egyik tömörítés tartalmazza a másikat), akkor abban az esetben amikor a rövidebb tömörítéshez adjuk ezt a mondatot, akkor a hozzáadott érték láthatóan nagyobb, mint abban az esetben, amikor a hosszabb tömörítéshez adjuk, mivel előfordulhat, hogy az új mondat által hordozott információkat azok a mondatok is tartalmazzák, amelyek nem részei a rövid tömörítésnek, de részei a hosszúnak. Vagyis, ha A jelöli a hosszabb, B a rövidebb tömörítést ($B \subseteq A$), v pedig a szöveg egy mondata, akkor teljesül, hogy:

$$\mathcal{L}(A + v) - \mathcal{L}(A) \leq \mathcal{L}(B + v) - \mathcal{L}(B)$$

Szeretnék megemlíteni néhány módot, ahogyan definiálni szokás az $\mathcal{L}$ függvényt. Ha $w_{i,j}$ jelöli a hasonlóságot az i. és a j. mondat között, akkor a legegyszerűbb,

ha

$$\mathcal{L}_1(S) = \sum_{i \in V, j \in S} w_{i,j}$$

Az előző jelölést használva a következőképpen is definiálhatjuk az $\mathcal{L}$ függvényt:

$$\mathcal{L}_2(S) = \sum_{i \in V} \max_{j \in S} w_{i,j}$$

Ez az úgynevezett "facility location", azaz szolgáltató elhelyezési szubmoduláris függvény, amiről dolgozatom 4. fejezetében részletesebben szó esik.
A 3.1.3.-as részben bemutatott "koncepió-alapú" függvénnyel is kifejezhető az adott tömörítés információtartalma:

$$\mathcal{L}_3(S) = \sum_{i \in \Gamma(S)} c_i$$

Egy további lehetséges definíció:

$$\mathcal{L}_4(S) = \sum_{i \in V} \min\{\mathcal{C}_i(S), \alpha \mathcal{C}_i(V)\},$$

ahol a $\mathcal{C}_i$ a V részhalmazain értelmezett valós értékű, monoton növő, szubmoduláris függvény azt méri, hogy az adott tömörítés mennyire adja vissza az i. mondat által hordozott információkat, $0 \leq \alpha \leq 1$.

3.4.1. Tétel ([3])**.** *Az imént definiált $\mathcal{L}_i$ függvények $(i = 1, ..., 4)$ mindegyike monoton növő és szubmoduláris.*

Bizonyítás. Az, hogy $\mathcal{L}_1$ és $\mathcal{L}_3$ monoton növő és szubmoduláris, az egyből látszik, az $\mathcal{L}_2$ függvényről pedig később szó lesz. Annak bizonyításához, hogy $\mathcal{L}_4$ is az, ismét az 1.3.7.-es állításhoz nyúlhatunk vissza, ugyanis a $\min(x, y)$ a külső függvény (ahol $y \geq 0$ állandó) monoton növő konkáv, a belső függvényről pedig tudjuk, hogy monoton növő szubmoduláris. Fel kell használni továbbá azt is, hogy az összegzés megőrzi mind a szubmoduláris (1.3.2.), mind a monoton növekedő tulajdonságokat.

□

3.4.2. A változatosságot mérő függvény

A változatosságot jutalmazó $\mathcal{R}$ függvény helyett használható egy olyan függvény, mely az ismétléseket bünteti, de ez nem szerencsés, mivel ekkor a függvény nem monoton növő. Ebből kifolyólag a gyakorlatban hasznos függvény a következő:

$$\mathcal{R}(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}$$

A definiálás során használt P_i $(i = 1, ..., K)$ a V alaphalmaz egy partíciója, azaz a P_i-k páronként diszjunktak és az uniójuk a V. Továbbá r_i jelöli az i egyedüli értékét (pl. a haszon, ha az üres halmazhoz adjuk).

3.4.2. Tétel ([3])**.** *Az így definiált $\mathcal{R}$ függvény monoton növő szubmoduláris.*

Bizonyítás. Ismét lehet alkalmazni az 1.3.7.-es állítást, ugyanis a külső függvény a négyzetgyök függvény, ami konkáv és monoton növő, a belső függvény pedig moduláris és mivel az r_i-k nemnegatívak, ezért monoton növő is. Az 1.3.7.-es állítás szerint az így kapott kompozíciófüggvény monoton növő és szubmoduláris, és mivel a nemnegatív lineáris kombináció képzés megőrzi ezeket a tulajdonságokat, ezért $\mathcal{R}$ monoton növő szubmoduláris függvény.

□

Megemlíthető, hogy a négyzetgyök függvény helyett lehet más monoton növő konkáv függvényeket is használni.

4. Apricot - Tanulási teszthalmazok

Az Apricot egy Python csomag, aminek segítségével hatalmas adathalmazból ki lehet választani egy olyan részhalmazt, ami az egész adatsokaságot reprezentálja. Ennek a legfontosabb felhasználása a gépi tanulás során lelhető fel, ugyanis nagyon hasznos, ha a tanulási folyamat során használt teszthalmaz számossága leredukálható úgy, hogy az elért eredmény alig változik. A kulcsgondolat az, hogy az adathalmaz növekedésével az ismétlések száma is nő, és ezektől az ismétlődésektől szeretnénk megszabadulni, mivel ezek már semmi előnyt nem jelentenek. Nem meglepő, hogy mindezt szubmoduláris függvények maximalizálásával lehet elérni, amihez az Apricot egy hatékony mohó algoritmust használ [9]. Kétféle fő függvénnyel dolgozik a program: az egyik az úgynevezett feature-based típusú függvény, amit magyarul tulajdonság-alapú függvénynek nevezünk, a másik pedig az úgynevezett facility location, azaz szolgáltató elhelyezési függvény, amiről már az előző fejezetben is szó esett. A továbbiakban ezen függvények előnyeiről és hátrányairól írok, majd ezeket különböző példákkal szemléltetem.

4.1. Szolgáltató elhelyezési függvény

A szolgáltató elhelyezési függvény [9] a nevét a híres szolgáltató-, másnéven vállalat elhelyezési feladatról kapta, vagyis arról a problémáról, amivel a cégeknek kell szembenézniük, amikor új létesítményt szeretnének nyitni és keresik ennek a helyét úgy, hogy ne legyen túl közel a már meglévő, hasonló létesítményekhez. Az ilyen függvény a következőképpen formalizálható:

$$\mathcal{F}_{szolg.}(S) = \sum_{i \in V} \max_{j \in S} w_{i,j},$$

ahol V az alaphalmaz, $S \subseteq V$ halmaz és w hasonlósági függvény úgy, hogy a $w_{i,j}$ az i-edik és j-edik elem közti hasonlóságot méri. A jelen esetben leggyakrabban használt hasonlósági függvények közé tartozik a negatív euklideszi távolság, illetve a koszinusz távolság. Az így definiált függvény szubmoduláris, ugyanis teljesíti a csökkenő hasznok elvét (az 1.1.2. állítást), mivel $S_2 \subseteq S_1 \subseteq V$ és $v \in V \setminus S_1$ esetén a v S_1-hez hozzáadott értéke nem nagyobb, mint a v S_2-höz adott értéke, mivel

lehetséges, hogy egy a v-hez hasonló mondat szerepel az $S_1 \setminus S_2$-ben.
Az Apricot a szolgáltató elhelyezési függvények maximalizálását egyszerű mohó algoritmussal végzi, melynek során a V alaphalmazt szeretnénk reprezentálni egy S^* részhalmazzal, melyre $|S^*| \ll |V|$. Ez a maximalizálási folyamat azonban nagy adathalmazon elég lassú is lehet, mivel az egyes mondatok közötti hasonlóságokat egy $n \times n$-es mátrix elemeinek feleltethetjük meg (ahol n a mondatok számát jelöli), és ezeket az elemeket mindig újra kell kalkulálni, amikor hozzáveszünk egy elemet a részhalmazhoz. Tehát az algoritmus minden iterációban $\mathcal{O}(n^2)$ lépést tesz. Ezen persze olykor programozási trükkökkel segíteni lehet. Egy további gyorsítási lehetőség az, ha a függvényt a következő alfejezetben szereplő, gyorsabban kiértékelhető függvényre cseréljük.

4.2. Tulajdonság-alapú függvény

Az előzőekhez hasonlóan a V legyen az alaphalmaz és $S \subseteq V$ halmaz. Továbbá definiáljunk különböző tulajdonságokat, amelyek meglétét ellenőrizzük az egyes mondatokra. A tulajdonságok számát jelöljük D-vel, az egyes tulajdonságokat pedig d-vel. A tulajdonságokhoz tartozó súlyt, azaz azt, hogy egy d tulajdonság mennyire fontos, jelölje c_d, a d tulajdonság értékét egy adott $s \in S$ mondatra jelölje s_d. Fontos feltenni, hogy a tulajdonságok értékei, illetve a tulajdonságokhoz tartozó súlyok nem vehetnek fel negatív értékeket, mert ekkor nem garantált a szubmodularitás. Ezen kívül még szükséges egy Φ monoton konkáv függvény is, melyre gyakran használt példa a négyzetgyök, illetve a logaritmus függvény. Ekkor a tulajdonság-alapú függvények (angolul feature-based függvények) [9] definiálhatók úgy, hogy

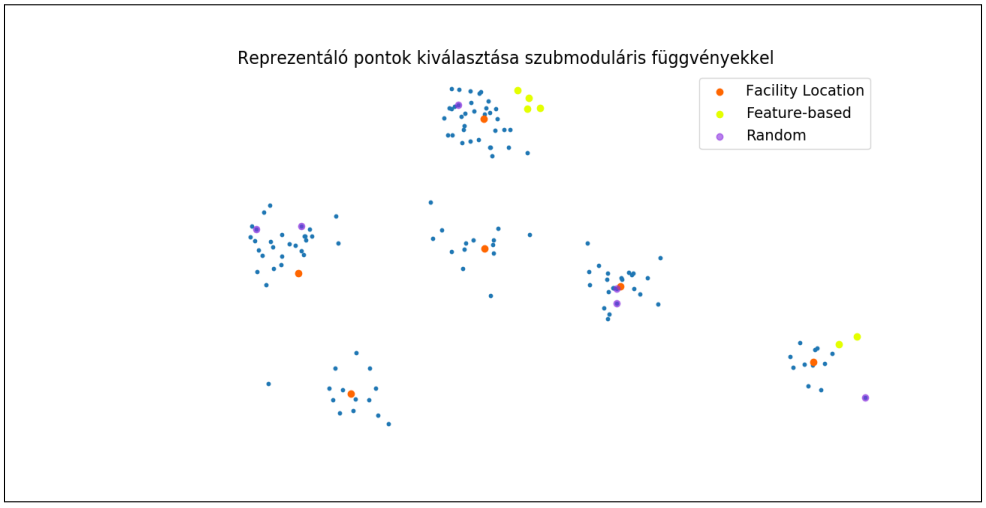
$$\mathcal{F}_{tul.}(S) = \sum_{d=1}^{D} c_d \Phi(\sum_{s \in S} s_d)$$

Ezzel a formalizálással is egy szubmoduláris függvényt kaptunk, ugyanis a Φ konkávitása, illetve a $\sum_{s \in S} s_d$ szubmodularitása miatt használható az 1.3.7.-es állítás, miszerint a $\Phi(\sum_{s \in S} s_d)$ szubmoduláris, és mivel a szubmoduláris függvények zártak a nemnegatív lineáris kombináció képzésre, ezért kapjuk, hogy az $\mathcal{F}_{tul.}$ függvény szubmoduláris.

Az $\mathcal{F}_{tul.}$ függvénynek egy speciális esete a 3.4.2.-es részben definiált változatosságot mérő függvény: a tulajdonságok maguk a partíciók, $c_d = 1$ az összes partícióra, Φ a négyzetgyökfüggvény.
Vegyük észre, hogy a tulajdonság-alapú függvényekkel a mohó algoritmus minden lépésének futási ideje $\mathcal{O}(Dn)$, amely jelentősen gyorsabb, mint a szolgáltató elhelyezési függvény során adódó $\mathcal{O}(n^2)$.

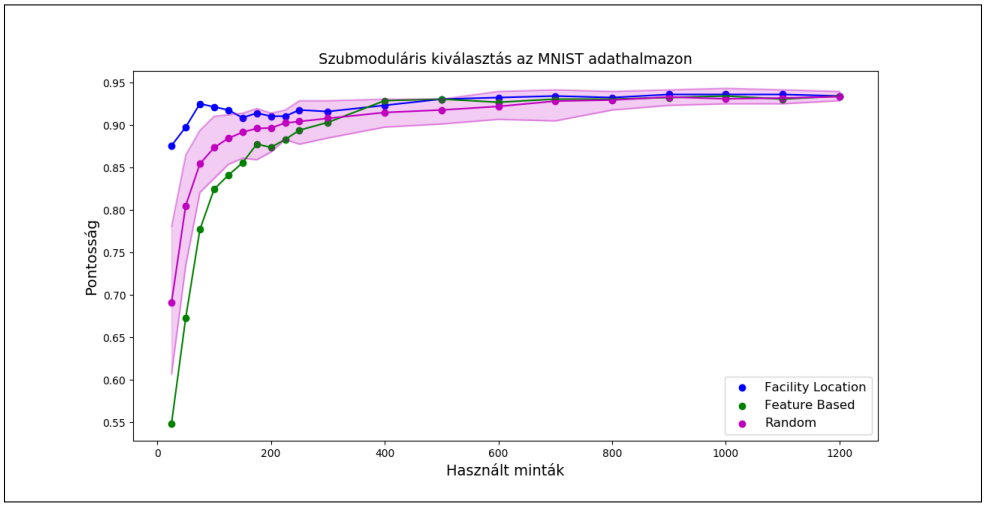
4.3. Példák

Az, hogy az előző két bekezdésben ismertetett szubmoduláris függvények közül melyiket célszerű használni, a megoldandó feladattól függ. Mindkét függvénynek vannak előnyei és hátrányai is, melyeket három példán keresztül szeretnék bemutatni.
Az első példában hat darab, különböző normális eloszlás által generálunk pontokat, összesen 140-et, és ezen pontok közül szeretnénk kiválasztani hatot úgy, hogy minél jobban reprezentálják a pontok eloszlását. Ezt háromféleképpen tesszük meg: 1. az $\mathcal{F}_{szolg.}$ szubmoduláris függvény maximalizálásával, 2. hasonlóan az $\mathcal{F}_{tul.}$ függvényt használva és 3. véletlenszerűen, random hat pontot választva. Az így kapott eredmények olvashatóak le az 1. ábráról:



1. ábra.

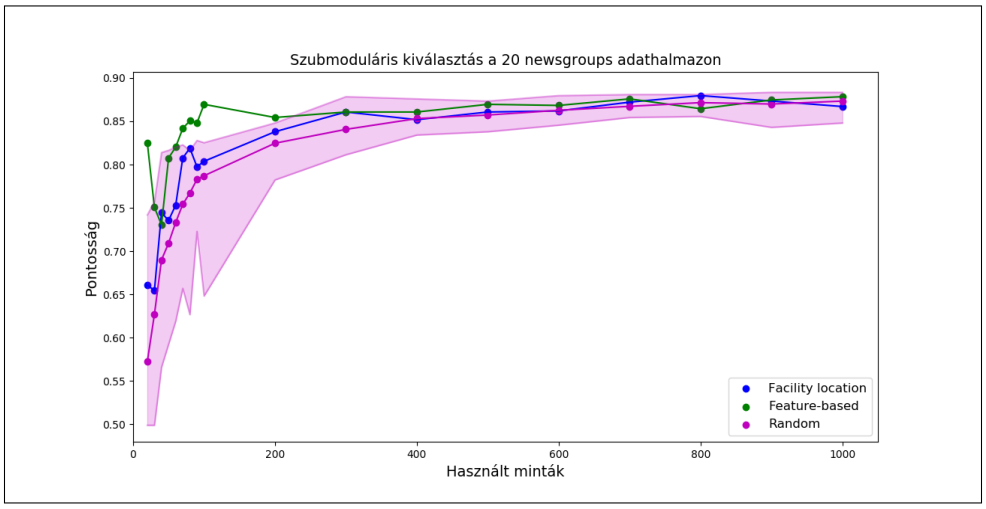
Megállapíthatjuk, hogy ezen feladatra a legjobb eredményt a szolgáltató elhelyezési függvény használata során kapjuk. A függvényhez tartozó hasonlósági függvény, a negatív euklideszi távolság reprezentatív a pontokra nézve, és így a k-közép klaszterezésnek egy jó közelítését kaphatjuk. A tulajdonság-alapú függvény viszont nem hozza a várt eredményt, ugyanis azon pontokat választja ki, melyeknek a tulajdonság értéke a legnagyobb. A jelen esetben alkalmazott tulajdonság az origótól való távolság, ami így nem reprezetálja az adathalmaz egészét, hanem csak a "határát". Mindezért fontos, hogy milyen tulajdonságokat definiálunk az adott adathalmazon, mert nem mind vezet reprezentatív redukcióhoz: A tulajdonság alapú függvények akkor működnek jól, ha a tulajdonságokhoz tartozó értékek szoros kapcsolatban vannak azzal, hogy az adott elem (jelen esetben mondat) mennyire fontos. Egy pénzügyi példa: egy ember minél több pénzt fektetett be egy értékpapírba, az annál fontosabb neki, tehát ha ebben az esetben a tulajdonságnak a pénz mennyiségét vesszük, akkor jó eredményt fogunk kapni. Nem jó tulajdonság például az autó színe, mert az nincs kapcsolatban annak értékével.
A következő példa során is a szolgáltató elhelyezési függvénnyel értük el a legjobb eredményt. Ezt olvashatjuk le a 2. ábráról:



2. ábra.

A feladat az volt, hogy minél kevesebb mintát használva tanítsuk meg a gépet

számjegyek felismerésére (az MNIST adathalmazon), azaz a cél a tanulási folyamat gyorsítása volt úgy, hogy az eredményen lehetőleg minél kisebb mértékben változtassunk. Azt, hogy a tulajdonság-alapú függvény miért nem működik erre a feladatra, ismét a nem megfelelő tulajdonsággal lehet indokolni. A függvény pixel kitöltöttségi értékekkel dolgozik, ezért a maximalizáló mohó algoritmus olyan mintákat választ, melyeknek a legnagyobb a kitöltöttsége, de ez nem eredményezi azt, hogy a minták különböző számjegyeket reprezentálnak. Ezzel szemben viszont a szolgáltató-elhelyezési függvény nagyon jól működik úgy, hogy négyzetes korrelációt használ, mivel az azonos számjegyeket tartalmazó képek egymáshoz jobban hasonlítanak, mint a más számjegyekről készült képekhez.
A harmadik példa pedig azt szemlélteti, hogy olykor a tulajdonság-alapú függvényekkel érhető el a legjobb eredmény. A 20 newsgroups adathalmazát használva arra szeretnénk megtanítani a gépet, hogy ismerje fel, melyik cikk szól a gyógyszerekről és melyik az űrről. Ehhez 1187 mintát használunk a tanítási folyamat során és 790-et a teszteléshez. A tanítási folyamat során használt minták számát szeretnénk csökkenteni, melynek eredményét láthatjuk a 3. ábrán:



3. ábra.

A grafikont elnézve szembetűnő, hogy a tulajdonság-alapú függvénnyel értük el a legjobb eredményt, olyannyira, hogy segítségével már néhány darab, körülbelül 100

mintával is majdnem ugyanazt azt eredményt sikerült produkálni, mint az 1187-tel. Ezt arra lehet visszavezetni, hogy az úgynevezett TF-IDF (term frequency-inverse document frequency) statisztikai eljárással súlyozzuk a szavakat. Ennek a működése két fázisból áll: az első során az adott szó gyakoriságát, előfordulását mérjük az őt tartalmazó szövegben, a második fázisban pedig azt, hogy az adott szó milyen gyakran, vagy milyen ritkán fordul elő a betáplált szövegekben. Az eljárás célja, hogy ne azt a szót nevezzük kulcsszónak, ami csak annyit teljesít, hogy sokszor szerepel egy szövegben, hanem azt, amelyik azt is teljesíti, hogy a többi szövegben ritkán fordul elő. Ha a tulajdonság-alapú függvényünk ezen tulajdonságot használja, akkor a maximalizálása során azon szövegeket fogjuk megkapni, amik a legtöbb kulcsszót tartalmazzák, ahol ezek a kulcsszavak tényleg visszaadják az adott szöveg lényegét.

5. Összegzés

Dolgozatomban megkíséreltem betekintést nyújtani abba, hogy a lényegkiemelési feladatokhoz milyen szubmoduláris modelleket szoktak társítani, ezekhez milyen, a gyakorlatban alkalmazható algoritmusokat lehet használni. Összességében kijelenthető, hogy a szubmodularitás természetesen jelen van a mesterséges nyelvfeldolgozási feladatokban. Ezen állítást támasztja alá a következő táblázat, mely az előző fejezetekben elhangzott feladatokat és a hozzájuk tartozó szubmoduláris függvényeket tartalmazza.

Szubmodularitás a nyelvfeldolgozásban		
Feladat	Eljárások	Függvények
Tömörítés adott felső korláttal	MMR; Módosított mohó algoritmus; Koncepció alapú tömörítés	$\mathcal{F}_{mmr}$; $\mathcal{F}_{cut}$, $\mathcal{F}^*_{cut}$; $\mathcal{F}_{concept}$
Tömörítés adott alsó korláttal	Minimális dominános halmaz ker.; Wolsey algoritmus;	$\mathcal{F}_{dom}$
Automatikus értékelés	ROUGE-N	$\mathcal{F}_{ROUGE-N}$
Tanulási folyamat redukálása	Mohó algoritmus	$\mathcal{F}_{szolg.}$, $\mathcal{F}_{tul.}$
További célfüggvények		$\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$; $\mathcal{R}$

Irodalomjegyzék

[1] Frank András, Összefüggések a kombinatorikus optimalizálásban II. - Szubmoduláris optimalizálás és poliéderes kombinatorika, Operációkutatási Tanszék és MTA-ELTE Egerváry Kutatócsoport, Eötvös Loránd Tudományegyetem, Budapest (2009)

[2] L. Lovász, Mathematical Programming The State of the Art, Submodular functions and convexity, *Springer*, (1983), 235-257

[3] Hui Lin, Jeff Bilmes, A Class of Submodular Functions for Document Summarization, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, Oregon, USA, (2011), 510–520

[4] Jaime Carbonell, Jade Goldstein, The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries, *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, Association for Computing Machinery*, New York, United States, (1998), 335-336

[5] Hui Lin, Jeff Bilmes, Multi-document Summarization via Budgeted Maximization of Submodular Functions, *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California, (2010), 912–920

[6] Laurence A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica*, (1981), 385–393

[7] Chao Shen and Tao Li - Multi-Document Summarization via the Minimum Dominating Set, *Proceedings of the 23rd International Conference on Computational Linguistics, Coling 2010 Organizing Committee*, Beijing, China, (2010), 984–992

[8] Chin-Yew Lin, ROUGE: A Package for Automatic Evaluation of Summaries *Text Summarization Branches Out, Association for Computational Linguistics*, Barcelona, Spain, (2004), 74–81

[9] Jacob Schreiber, Jeffrey Bilmes, William Stafford Noble, Apricot: Submodular selection for data summarization in Python, (2019)

[10] Alexander Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *Journal of Combinatorial Theory, Series B, Volume 80, Issue 2*, (2000), 346-355

[11] Maxim Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, *Operations Research Letters 32*, IBM, T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA, (2004) 41 – 43

[12] Tatsunori Mori, Masanori Nozawa, Yoshiaki Asada, Multi-answer-focused multi-document summarization using a question-answering engine *ACM Transactions on Asian Language Information Processing Vol. 4, No. 3*, Yokohama, Japan (2005)

[13] http://swoh.web.engr.illinois.edu/courses/ie512/handout/submodular.pdf