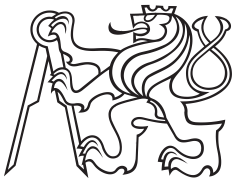
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Efektivní výpočet podobnosti sekundárních struktur RNA

Bc. Marek Hrvol

Vedoucí: Doc. Ing. Jiří Kléma, Ph.D.
Obor: Otevřená informatika
Studijní program: Bioinformatika
Květen 2019

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hrvol** Jméno: **Marek** Osobní číslo: **474540**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Bioinformatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Efektivní výpočet podobnosti sekundárních struktur RNA

Název diplomové práce anglicky:

Efficient similarity calculation for RNA secondary structures

Pokyny pro vypracování:

1. Seznamte se s funkcí a strukturou ribonukleových kyselin (RNA).
2. Vypracujte rešerši stávajících metod pro výpočet podobnosti mezi sekundárními strukturami RNA.
3. Z veřejných zdrojů získejte knihovnu sekundarních struktur RNA. Pracujte s ověřenými strukturami, zvažte využití struktur predikovaných.
4. Navrhněte vlastní metodu efektivního výpočtu podobnosti mezi sekundárními strukturami, zvažte analogii přístupu doc2vec aplikovanou na struktury namísto dokumentů.
5. Srovnejte svůj přístup s přístupy nalezenými v literatuře. Srovnávejte z hlediska shody v podobnosti i z hlediska rychlosti.
6. Svůj přístup aplikujte v konkrétní úloze vyhledávání virových strukturních motivů ve vybraných částech lidského transkriptomu.

Seznam doporučené literatury:

Na, L. and Wang, T. 'A method for rapid similarity analysis of RNA secondary structures.' BMC Bioinformatics 7.1 (2006): 493.
Le, Q. and Mikolov, T. 'Distributed representations of sentences and documents.' In Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 1188–1196, 2014.
Gardner, P. P. and Giegerich, R.. 'A comprehensive comparison of comparative RNA structure prediction approaches.' BMC Bioinformatics, 5(1):140, 2004.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jiří Kléma, Ph.D., Intelligent Data Analysis FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **22.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

doc. Ing. Jiří Kléma, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Rád bych poděkoval všem, kteří mě podporovali při psaní této diplomové práce. Z akademické půdy je to především vedoucí mé diplomové práce docent Jiří Kléma, který mi poskytoval odborné konzultace, rozšířil mé znalosti strojového učení a pomáhal odhalovat nedostatky mé diplomové práce. Další lidé z akademické půdy, kterým bych chtěl poděkovat jsou všichni učitelé, kteří mě za dobu studia učili. Věřím, že mé znalosti se za poslední dva roky silně rozšířily právě díky vám. Také bych chtěl poděkovat Markovi Zvarovi a dalším spolužákům, se kterými jsme si mnohdy dovysvětlovali látku a sdíleli užitečné studijní materiály.

V neposlední řadě bych chtěl poděkovat lidem, od kterých se mi dostávalo největší podpory nejen při psaní diplomové práce, ale po celou dobu studia – své rodině – sestře, bratrovi a především rodičům za podporu při studiu, za kterou mohu být doopravdy vděčný. Nakonec bych chtěl poděkovat své přítelkyni Paulíně Strečanské, která mi byla psychickou podporou ve všech náročných situacích během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 23. května 2019

Abstrakt

Tato práce se zaobírá porovnáváním sekundárních struktur RNA pomocí mělkých neuronových sítí. Název nově vytvořené metody je Struc2Vec.

Způsob výpočtu je založen na přepisu sekundárních struktur na slova, která jsou následně přepsána na vektory pomocí technologie Doc2vec. Pro fungování metody je nutné natrénovat model na strukturách, který lze poté využít k porovnávání s jinými strukturami. Trénování i predikce mají lineární časovou náročnost.

Metoda dosáhla skoro 74% úspěšnosti při predikci mezi deseti typy struktur a skoro 96% úspěšnosti při porovnávání mezi dvěma typy sekundárních RNA struktur (eukaryoty a bakterie). Úspěšnost použitím klasických metod na stejných data-setech je 85 %, respektive 98 %.

Metoda Struc2Vec nabízí alternativní možnost k porovnávání sekundárních struktur, která je vhodná v případě, že je potřeba zkrátit čas výpočtu porovnávání sekundárních struktur RNA.

Klíčová slova: Struc2Vec, neuronové sítě, Doc2vec, predikce, sekundární struktura, RNA, vnoření slov, uměla inteligence, strojové učení

Vedoucí: Doc. Ing. Jiří Kléma, Ph.D.
Karlovo náměstí 13,
E-432,
Praha 2

Abstract

This work focuses on comparison of secondary RNA structures using shallow neural networks. Name of newly developed method is Struc2Vec.

Struc2Vec method is based on transforming secondary RNA structures into words, which are further transformed into vectors. The method can work only after training a model with secondary structures. The model is further used for comparing different secondary structures with already trained ones. Both model training and structure prediction work in linear time complexity.

The method had nearly 74 % success rate on dataset with 10 different types of secondary structures and nearly 96 % on dataset of two types of structures. Compared to the standard methods, which had 85 %, respectively 98 % success rate.

Struc2Vec method offers very good trade-off between speed and accuracy. Method is suitable for cases where computation time needs to be shortened significantly.

Keywords: Struc2Vec, neural network, Doc2vec, prediction, secondary structure, RNA, word embedding, artificial inteligence, machine learning

Title translation: Efficient similarity calculation for RNA secondary structures

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Tato práce se zaobírá porovnáváním sekundárních struktur RNA pomocí umělé inteligence, zejména mělkých neuronových sítí. Motivací pro porovnávání vyššího stupně RNA struktur je to, že biologická funkce RNA je velmi často určena její vyšší strukturou. Ani vysoký počet mutací na úrovni primární struktury, tedy prosté sekvence nukleotidů, se nemusí nutně promítnout do změny vyšší struktury a tím ani změny funkce. K posouzení homologie páru RNA, a tím i jejich funkční příbuznosti, pak přímé porovnání jejich řetězců nestačí. I řetězce s velmi nízkou shodou mohou být homologické. Zejména u rychle mutujících sekvencí, jako například u virů, je nutné provést jejich porovnání až po zavinutí do vyšší struktury.

V současné době existuje více přístupů pro porovnávání sekundárních struktur RNA, avšak přístupy poskytující dobré výsledky pracují v kubické časové náročnosti a porovnávají pouze dvě struktury najednou. Při porovnávání velkého množství dlouhých struktur může doba výpočtu nabývat obrovských hodnot. Další nevýhodou některých přístupů pro porovnávání sekundárních struktur je zohledňování primární struktury, což může být nevýhodou, pokud primární struktura porovnávané RNA podlehla velmi vysokému počtu mutací.

Tato práce přichází s inovativním způsobem, jak nalézt podobnost mezi jednotlivými sekundárními strukturami RNA. Cílem je vytvoření heuristického nástroje, který dokáže porovnávat sekundární RNA struktury v lineárním čase a zároveň dokáže porovnat i více struktur najednou.

Biologickou motivací pro tuto práci je porovnávání sekundárních struktur RNA virů (například vir Hepatitidy C) se sekundární strukturou RNA člověka.

Lze očekávat, že strukturní motivy využívané virovými RNA pro interakci s hostitelem se mohou konzervovat i v transkriptomu hostitele. Příkladem RNA, která zachovává svou sekundární strukturu je IRES (z angl. internal ribosome entry site), která umožňuje virům za jistých podmínek výrazně ovlivnit syntézu bílkovin hostitele.

Jedním z největších současných problémů pro řešení tohoto problému je délka RNA člověka. Pro představu, délka lidského genomu je přibližně 3 000 000 000 bazických párů [59] a doba porovnání dvou struktur o 2 000 bazických párech trvala pomocí klasických metod přibližně 4 sekund. Pokud by byl lidský genom rozdělen po 2 000 bazických párech, vzniklo by přibližně 1 500 000 struktur. Pro porovnání všech těchto struktur pomocí tradiční metody by bylo potřeba porovnat každou strukturu s každou. Doba porovnání by byla přibližně $1499999! \cdot 4$ sekund. Tento odhad je samozřejmě výrazně nadhodnocen, protože v reálném případě by se porovnávali pouze vybrané části genomu.

Biologům by výsledný nástroj mohl umožnit vyhledávání kandidátních výskytů motivů interního vazebního místa pro ribozom virů v transkriptomu lidské buňky. Výsledný nástroj může také sloužit k dalšímu studiu lidské RNA na úlohách, kde je potřeba porovnávat sekundární struktury RNA. Takovou úlohou by mohlo být například zkoumání změny sekundární struktury při změně primární struktury.

Část I

Teoretická část

Kapitola 2

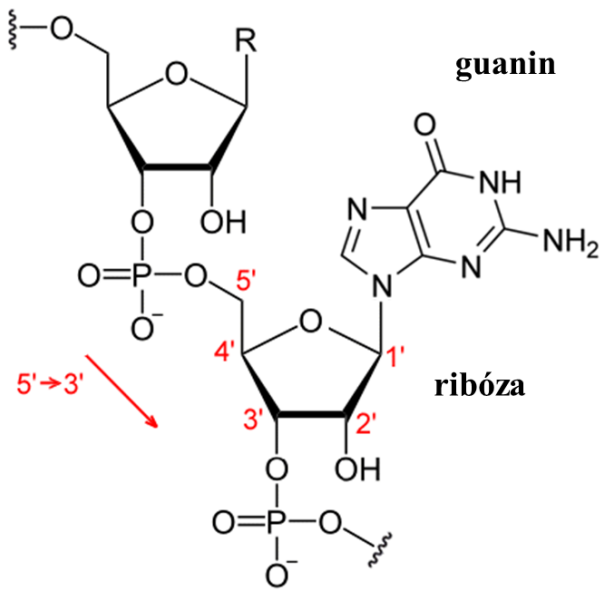
RNA

Tato kapitola se zabývá popisem ribonukleových kyselin, které jsou většinou známé pod názvem RNA (DNA v případě deoxyribonukleových kyselin, které se mění (transkribují) na RNA [60]). Pro pochopení této diplomové práce je užitečné se seznámit alespoň se sekundární strukturou RNA, avšak obecná znalost RNA může také pomoci.

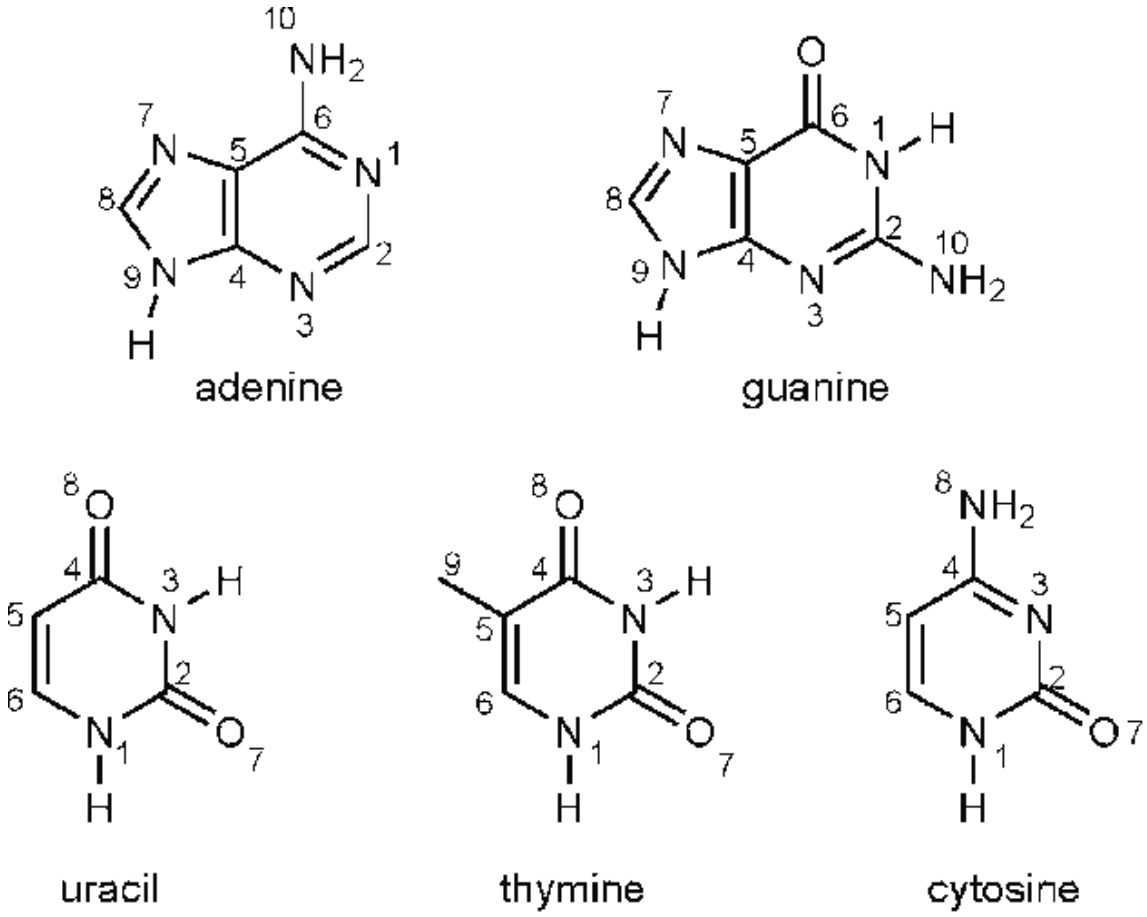
Ribonukleová kyselina (RNA) je tvořená vláknem ribonukleotidů. Ribonukleotid je složen z báze, ribózy a fosfátové skupiny. Jednotlivé složky tohoto vlákna jsou spojené fosfodiesterovou vazbou, která směruje od hydroxylu v poloze 3' ribózi k hydroxylu na 5' ribóze. Vlákno má tedy oba své konce rozdílné, což způsobuje polaritu řetězce. Při navazování nových bází je využíván 3' konec, tedy nové báze se navazují ve směru 5' -> 3' [32]. Báze jsou navázány na ribózu pomocí β-N-glykosidové vazby.

Báze lze rozdělit na dvě skupiny dle jejich hlavní části, která je vždy složena z aromatické heterocyklické organické sloučeniny – pyrimidin nebo purin. Kruh pyrimidinu obsahuje šest jednotek, dusík se nachází na pozicích 1 a 3 a uhlík na ostatních pozicích[1]. Pyrimidinové báze nacházející se v nukleových kyselinách jsou cytosin, thymin (pouze DNA) a uracil (pouze RNA). Purin je tvořen z pyrimidinového a imidazolového kruhu. Purinové báze nacházející se v nukleových kyselinách jsou guanin a adenin[2].

[1] https://en.wikipedia.org/wiki/Pyrimidine
[2] https://en.wikipedia.org/wiki/Purine



Obrázek 2.1: Strukturní vzorec RNA [42].



Obrázek 2.2: Báze nukleových kyselin [34].

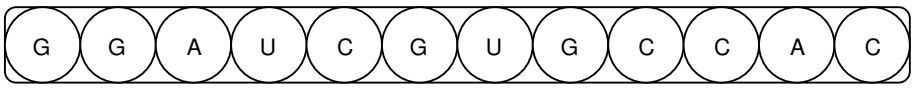
2.1 Struktury RNA

Popis struktury RNA lze rozdělit do tříd podle toho jak moc informací o struktuře RNA popisují. V této sekci se zaměřujeme tyto popisy struktur. Začneme od nejzákladnějšího popisu, tedy od primární struktury a dostaneme se až ke kvartérní struktuře RNA. Pro tuto práci je nejdůležitější mít znalosti o sekundární struktuře RNA, jelikož se v této diplomové práci bude hojně využívat. Sekundární struktura je zde rozebírána nejpodrobněji.

2.1.1 Primární struktura

Primární struktura popisuje lineární rozložení daného řetězce. Primární struktura se typově popisuje stejným způsobem u RNA, DNA a také u proteinu (rozdíl existuje pouze v názvu jednotlivých částí struktury). Při zápisu primární struktury je důležité pouze pořadí bází. Je tedy bráno v potaz pouze spojení se sousedy v 1D a informace o jakémkoliv jiném spojení, které by se mohlo ve vyšších dimenzích (například 2D) není dostupná [41].

U RNA je tedy primární struktura definována řetězcem skládajících se ze znaků abecedy $\Sigma = \{A, C, G, U\}$, kde každé písmeno reprezentuje bázi příslušného nukleotidu.



Obrázek 2.3: Vizualizace primární struktury RNA[3].

Primární strukturu RNA lze díky její jednoduchosti zapsat více způsoby (například na obrázku zobrazený způsob pouhého řetězce). Avšak jedním z velmi častých způsobů zápisů je formát FASTQ. Tento formát má oproti jiným formátům (například oproti prostému řetězci znaků) výhodu, že kombinuje sekvenci s kvalitou čtení dané báze [9].

FASTQ formát se skládá z popisu částí struktury, každá část je popsána pomocí čtyř řádků. První řádek začíná znakem "@"a obsahuje identifikátor jednotlivé části a případně nepovinný popis. Druhý řádek samotnou sekvenci. Třetí začíná znakem "+"a často obsahuje stejný identifikátor (případně popis) jako první řádek, avšak tyto informace jsou v třetím řádku nepovinné. Čtvrtý řádek obsahuje symboly značící kvalitu čtení sekvence nukleotidů v druhém

[3] Vytvořeno v aplikaci https://www.draw.io/

Listing 2.1: Příklad formátu FASTQ

```
@READ_240
AGAGAAUCA
+
A9A::99B?
```

řádku. Čtvrtý a druhý řádek musí obsahovat stejný počet znaků. Kvalita čtení sekvence nukleotidů v tomto kontextu znamená pravděpodobnost, že sekvence byla správně určena. Symboly na čtvrtém řádku nabývají hodnot od 0x21 (znak "!") do 0x7e (znak "∼"), kvalita roste vzestupně. Příklad FASTQ formátu lze vidět na obrázku 2.1.

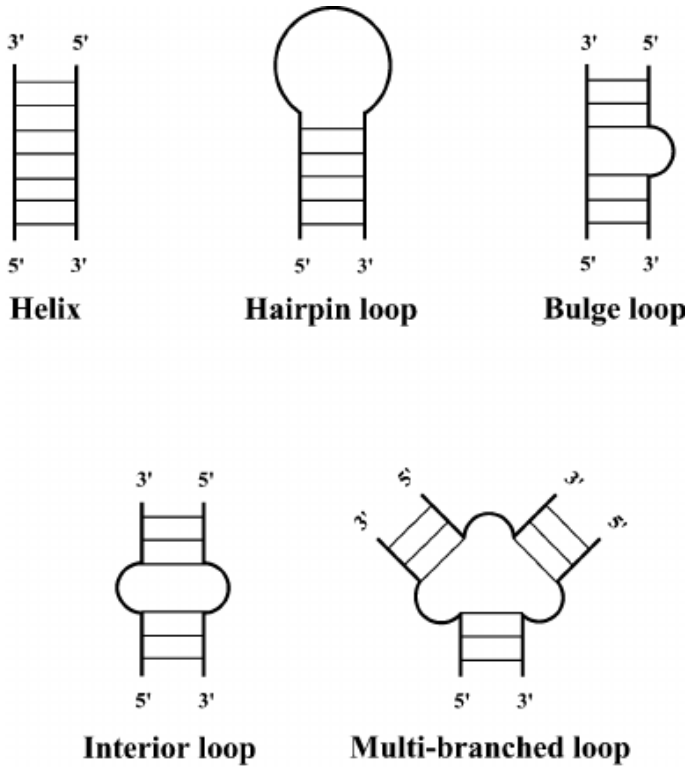
2.1.2 Sekundární struktura

Na začátku této sekce je popsáno, jak vypadá sekundární struktura RNA a jak vypadají její základní motivy. Ve druhé části jsou popsány formáty zápisu sekundárních struktur RNA, které budou používány během diplomové práce.

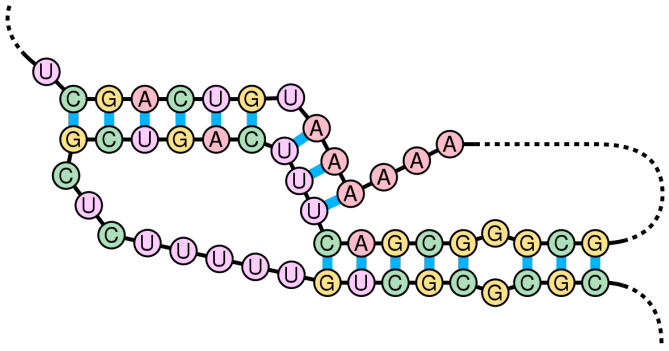
V biologii se pár nukleotidů, který je propojen vodíkovou vazbou nazývá bazický pár. Nejběžnějším způsobem párování v RNA je guanin s cytosinem a uracyl s adeninem. Tomuto párování se říká Watson-Crick[4] párování. Různé utváření těchto bazických páru způsobuje vznik různých motivů v sekundární struktuře RNA, která popisuje interakci bází v rámci jednoho polymeru nukleových kyselin [23]. RNA sekundární struktura se nejčastěji skládá z níže uvedených motivů [14], které jsou vizualizovány na obrázcích 2.4 a 2.5.:

- vlásenka (z angl. stem-loop nebo také hairpin, či hairpin loop)
- vyboulená smyčka (z angl. bulge loop)
- vnitřní smyčka (z angl. internal loop nebo také interior loop)
- rozvětvená smyčka (z angl. multibranched loop)
- šroubovice (z angl. helix)
- pseudoknot

[4] https://teaching.ncl.ac.uk/bms/wiki/index.php/Watson-Crick_base_pairing



Obrázek 2.4: Příklad motivů sekundárních struktur RNA [14].



Obrázek 2.5: Příklad pseudoknotu ve struktuře RNA [45].

Listing 2.2: Příklad závorkového formátu

```
>ENERGY = 0 loop RNase P RNA
AGAGAAUCA
.((...)).
```

Formáty sekundární struktury

Tato sekce obsahuje informace o několika zápisech sekundárních RNA struktur. Tyto formáty jsou využívané v této diplomové práci.

- závorkový formát (angl. *dot-bracket notation* nebo také *Vienna format*, přípona *.db*) [26]
- propojovací formát (z angl. *connectivity table* nebo *connect-format*, přípona *.ct*). Jedná se o formát od Michaela Zuckera [43].
- sekvence párů bází (angl. *base-pair sequence format*, přípona *.bpseq*) [54].

Závorkový formát je notace pomocí řetězce pro RNA sekundární struktury. Báze, která není spárovaná s jinou bází je označena tečkou. Báze, která je spárována s jinou bází je označena závorkou – otevírací závorka značí, že báze, se kterou se aktuální báze pojí, následuje až po současné bázi, Uzavírací závorka znamená přesný opak. Problémem závorkového formátu je zápis pseudoknotu, který tato notace neumožňuje, pokud není rozšířena o více typů závorek [55]. Z tohoto důvodu jsou ostatní formáty zápisu v této diplomové práci preferované. Před samotným závorkovým formátem lze často vidět také hlavičku a primární strukturu RNA. Příklad závorkového formátu lze vidět na obrázku 2.2

Propojovací formát je sloupcový formát, který obsahuje šest sloupců. První sloupec označuje index současné báze. Druhý sloupec značí bázi, která se nachází na daném indexu, tato báze je označena pomocí jednoho písmena (primární struktura). Třetí, čtvrtý a šestý sloupce jsou nadbytečné – obsahují indexy následujících (respektive předcházejících a současných) indexů. Čtvrtý sloupec obsahuje index báze, se kterou je současná báze spárována. Pokud je ve čtvrtém sloupci hodnota 0, znamená to, že báze není spárována s žádnou jinou bází [55].

Propojovací formát obsahuje v prvním řádku hlavičku, která zpravidla obsahuje počet bází, volnou energii struktury, název organismu apod. Některé

Listing 2.3: Příklad propojovacího formátu

```
9 ENERGY = 0 loop RNase P RNA
1 A       0    2    0    1
2 G       1    3    8    2
3 A       2    4    7    3
4 G       3    5    0    4
5 A       4    6    0    5
6 A       5    7    0    6
7 U       6    8    3    7
8 C       7    9    2    8
9 A       8    0    0    9
```

Listing 2.4: Příklad formátu sekvence párů bází

```
Filename: file.bpseq
Organism: organism example
Accession Number: T00000
Citation and related information available at nowhere
1 A    0
2 G    8
3 A    7
4 G    0
5 A    0
6 A    0
7 U    3
8 C    2
9 A    0
```

programy očekávájí, že se v hlavičče bude nacházet heslo "ENERGY", případně "Energy"nebo "dG". Bez těchto slov je možné, že nebude soubor považován za validní [55]. Příklad tohoto formátu lze vidět na obrázku 2.3.

Formát sekvence párů bází je sloupcový formát (stejně jako propojovací formát). Tento typ obsahuje tři sloupce. První sloupec obsahuje pozici v sekvenci a je indexován od jedničky. Druhý sloupec obsahuje bázi nacházející se na daném indexu (primární struktura). Třetí sloupec obsahuje index spárované báze, v případě, že žádná báze není spárovaná je hodnota nulová. Tento formát také velmi často obsahuje hlavičku, která není povinná. Pokud je hlavička přítomna, je nutné, aby obsahovala název souboru, organismus, přístupové číslo a citaci [55]. Tento typ zápisu je typický pro data stažené z CRW databáze[5] [7] ("Comparative RNA web", obsahuje mnoho struktur RNA). Příklad tohoto formátu lze vidět na obrázku 2.4.

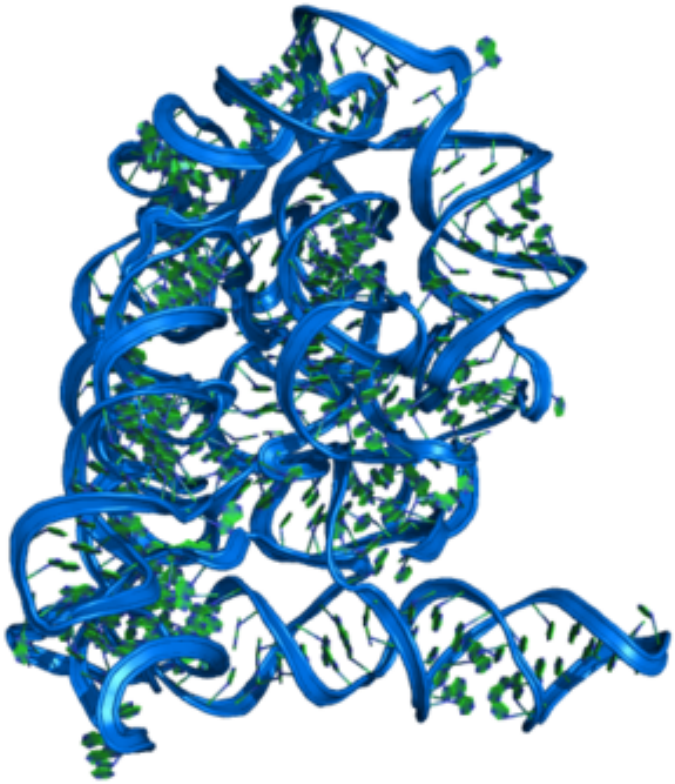
Ze zmíněných formátu jsou nejpoužívanější propojovací (*.ct*) formát a

[5] http://www.rna.icmb.utexas.edu/

závorkový formát (*.db*). Programy, které pracují se sekundárními strukturami RNA často podporují alespoň jeden z těchto dvou formátu jak na vstupu tak na výstupu. Jedna z největších výhod propojovacího formátu je jednoduchá práce s formátem v programech, jelikož lze načíst jako tabulka. Závorkový formát sice neposkytuje tak jednoduché načtení a má již zmíněný problém se zápisem pseudoknotu, ale na druhou stranu je velmi kompaktní, což při práci s velkým množstvím dlouhých struktur může ušetřit mnoho místa na disku [43].

2.1.3 Terciární a kvartérní struktura

Existují i popisy RNA na vyšší úrovni, než sekundární. Terciární struktura značí prostorové (3D) rozložení dané molekuly RNA (případně jiné makromolekuly). Oproti sekundární struktuře přibývá tedy další rozměr [35]. Kvartérní struktura značí ještě vyšší úroveň organizace. Jedná se o interakci mezi podjednotkami makromolekul (RNA, protein...) [35].



Obrázek 2.6: Katalytická molekula RNA s bohatou terciární strukturou (Oceanobacillus iheyensis). [44]

2.2 Mutace

Mutace je změna nukleotidové sekvence v genomu organismu. Kromě změny může nastat také vložení nové báze, smazání báze nebo duplikace nukleotidové sekvence. Mutace může proběhnout na jedné bázi DNA nebo může dojít ke větší mutaci, která ovlivní více genů na chromozomu [40].

Viry často využívají různé mechanismy úpravy své nukleotidové sekvence za účelem přežití. Viry se dokáží replikovat velmi rychle a jejich nukleotidové báze jsou často velmi zmutované. Díky velkému množství mutací mohou dva organismy s velmi odlišnými nukleotidovými strukturami reprezentovat stejný vir. Změny v nukleotidových bázích se nemusí nutně znamenat změnu ve vyšších strukturách (sekundární...) [15].

Kapitola 3

Současné přístupy porovnání sekundárních struktur RNA

Velká část funkcionality RNA je určena její sekundární strukturou. Z tohoto důvodu může porovnávání sekundárních struktur pomoct určit podobnosti ve vlastnostech různých RNA [37].

Tato kapitola se zabývá přístupy porovnávání sekundárních struktur RNA. V první sekci se k dané sekvenci přistupuje pouze jako k řetězci znaků. Další sekce popisuje typy porovnávání pomocí editace stromu, které se dnes používají v oblastech zpracování obrazu, automatického dokazování teorémů, bioinformatiky a dalších [62]. Na závěr této kapitoly jsou popsány ještě algoritmy založené na jiných principech a následuje shrnutí přístupů použitých pro porovnávání sekundárních RNA struktur.

3.1 Přístupy porovnávání řetezců

Cílem při porovnávání dvou řetězců je změřit, jak moc jsou si řetězce podobné. Tato sekce popisuje různé přístupy, jak změřit podobnost dvou řetězců.

3.1.1 Hammingova vzdálenost

Hammingova vzdálenost je jeden z nejjednodušších způsobů, jak porovnat dva různé řetězce. Tato metoda bere v potaz počet neshod mezi jednotlivými řetězci, tedy výsledek je minimální počet změn znaku prvního řetězce, aby se dosáhlo druhého řetězce. Pro tuto vzdálenost není možné mít řetězce různých délek [21]. Časová složitost tohoto algoritmu je $O_{time}(N + N) = O_{time}(N)$, kde N značí počet znaků v řetězci [46].

Hammingova vzdálenost při porovnání sekundárních struktur (závorkový formát) (.(.)) a .(().) je 4 – neshoda na pozicích 1, 2, 4 a 5 (indexováno od 1).

Tato metoda patří mezi jedny z jednodušších, avšak pro porovnávání sekundárních RNA sekvencí není tato metoda ideální z důvodů, že oba řetězce musí být stejné délky [46].

3.1.2 Levenshteinova vzdálenost

Levenshteinova vzdálenost je rozšíření Hammingovy vzdálenosti o operace vložení znaku a smazání znaku. Tato vzdálenost je mnohem vhodnější pro většinu typů úloh, protože již nemusí platit, že délky obou řetězců musí být totožné.

Zpravidla se počítá s tím, že každá operace má hodnotu 1 a výsledkem je tedy pouze počet operací. Avšak existují i možnost ohodnocení typu operace, například je možné nebrat v potaz změnu, ale pouze vložení nebo smazání znaku. V tomto případě by změna znaku měla váhu 2, protože se musí jednat o smazání původního znaku a následovné přidání jiného znaku [30].

Levenshteinova vzdálenost při porovnání struktur ((..)) a ((.)) je 1 – z první struktury stačí odebrat jednu tečku. Hammingova vzdálenost na tomto případě vypočítat nelze, kvůli rozdílné délce struktur.

Složitost výpočtu Levenshteinovy vzdálenosti je vyšší než u Hammingovy vzdálenosti a to $O_{time}(N \cdot M) = O_{time}(N^2)$ [30]. Případně složitost modifikace algoritmu, která běží v čase $O_{time}(N \cdot (\frac{s}{2} - \frac{(N-M)}{2})$, kde S je editační vzdálenost [5].

3.1.3 Bioinformatické algoritmy

V bioinformatice se využívají metody, které jsou založeny na velmi podobném principu jako Levenshteinova vzdálenost. Mezi tyto metody patří zejména metody zarovnávání dvou sekvencí. Výsledkem těchto metod tedy není pouze hodnota, jak jsou řetězce od sebe vzdálené, ale také informace, jak musí být vůči sobě zarovnané pro obdržení dané vzdálenosti. Pro globální zarovnání je využíván Needleman-Wunschův algoritmus [38], pro lokální zarovnání je využíván Waterman-Smithův algoritmus [56].

Složitost výše zmíněných algoritmů je závislá na penalizaci za mezery v sekvencích. Existuje více přístupů, jak ohodnotit chybějící nukleotid. Jeden z jednodušších přístupů je ohodnocení každého chybějícího nukletidu konstantní hodnotou [56]. V takovém případě je nejlepší dosažená složitost těchto algoritmů $O_{time}(\frac{M \cdot N}{log(n)})$ [57].

Pro ohodnocování chybějících nukleotidů lze však použít i metody, které ohodnocují více mezer po sobě rozdílně, než mezery, které po sobě nejdou. Pro obecné metody je časová složitost $O_{time}(M \cdot N^2)$ [56]. Existuje i rychlejší způsob, který má pracuje v čase $O_{time}(M \cdot N)$ [20]. Avšak tento algoritmus nemusí vždy nalézt optimální řešení [4].

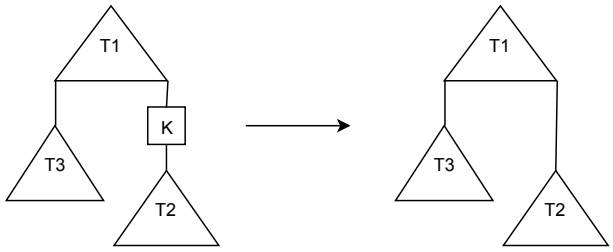
Algoritmy pro lokální zarovnávání mají paměťovou složitost $O_{space}(N \cdot M)$ [56]. Oproti tomu u globálního zarovnávání existuje metoda pro zlepšení prostorové složitosti na $O_{space}(min(N, M))$ [24].

3.2 Přístupy na základě editace stromů

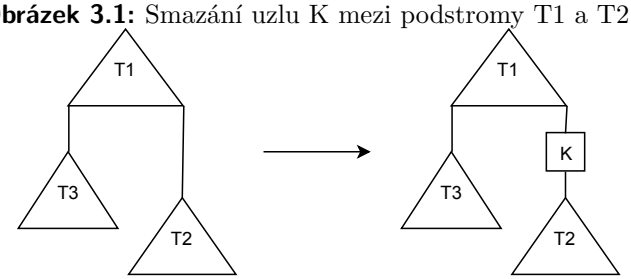
Tato sekce se zaměřuje na porovnání RNA sekundárních struktur pomocí editace stromů, což je v dnešní době jedna z velmi využívaných metod [17]. Na začátku této sekce je popis editační vzdálenosti stromů. Poté je ukázán algoritmus převodu sekundární RNA struktury do stromu. Následují různé existující algoritmy pro výpočet editační vzdálenosti a samotný popis implementace jednoho ze algoritmů pro výpočet editační vzdálenosti.

Vzdálenost vycházející z editace stromů uspořádaných označených stromů je dána nejmenší cenou sekvence základních editačních operací. Ty postupně transformují jeden strom do tvaru stromu druhého [28].

Mezi editační operace patří smazání uzlu (3.1), vložení uzlu (3.2) a nahrazení uzlu jiným uzlem. Každá operace může mít jiné ohodnocení (cenu), avšak vložení uzlu a smazání uzlu by mělo mít stejné ohodnocení, jelikož smazání uzlu v jednom stromu je totéž, jako přidání uzlu ve stromu druhém [13]. Díky této vlastnosti lze také porovnávat stromy pomocí transformace do navzájem izomorfních stromů pouze za pomoci operací vložení a nahrazení uzlu (případně smazání a nahrazení uzlu).



Obrázek 3.1: Smazání uzlu K mezi podstromy T1 a T2 [1].



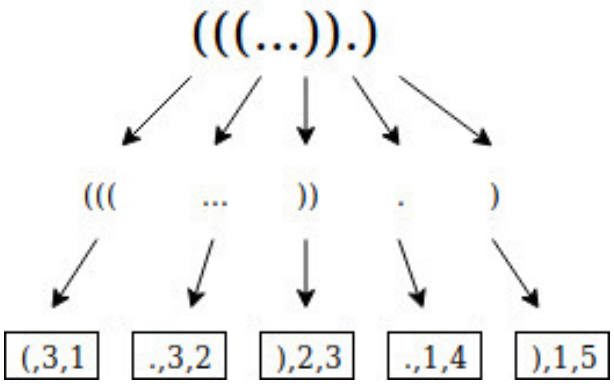
Obrázek 3.2: Přidání uzlu K mezi podstromy T1 a T2 [1].

3.2.1 Převod sekundární struktury RNA na strom

Algoritmus na převod závorkového formátu zápisu na strom lze popsat ve třech krocích. V prvním kroku je řetězec znaků převeden na list trojic, kde každá trojice obsahuje znak, počet po sobě jdoucích znaků daného typu a pořadí ve struktuře. Složitost této části transformace je $O_{time}(N)$), jelikož se provádí pouze jeden průchod řetězcem [16].

Druhá část algoritmu využívá struktury z první části. Výstupem této části algoritmu je zásobník obsahující jednotlivé uzly stromu v pořadí pre-order. Pro vytvoření zmíněného zásobníku využívá algoritmus dvou zásobníku, jednoho na uzly typu tečka a druhého na uzly typu závorka. Podle uzlů typu závorka dokáže algoritmus zjistit, na jaké úrovni se ve stromu nachází. Dokáže tak také zjistit kolik podstromů daný uzel obsahuje. Složitost této části transformace je $O_{time}(N)$), jelikož stačí jeden průchod přes všechny uzly (v nejhorším případě je nutno provést N operací nad oběma využitými zásobníky) [16].

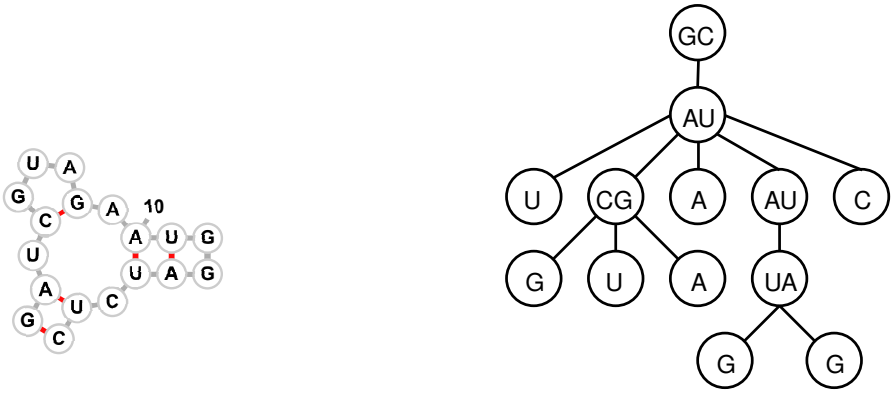
[1] Vytvořeno v aplikaci https://www.draw.io/
[2]Vytvořeno v aplikaci https://www.draw.io/



Obrázek 3.3: Trojice vytvořené v prvním kroku transformace [2]

Třetí část algoritmu pouze provede sken komponent ze zásobníku vytvořeného v druhé části algoritmu transformace a vybuduje odpovídající strom. Zásobník je potřeba projít pouze jednou, tedy složitost této části algoritmu je $O_{time}(N)$ [16].

Z popsaných částí algoritmu převodu řetězce ze závorkového formátu na strom lze určit, že převod má celkovou časovou složitost $O_{time}(N)$). Příklad takového převodu lze vidět na obrázku 3.4a.



(a) : Příklad RNA **(b) :** Graf RNA

Obrázek 3.4: Příklad RNA struktury transformované do stromu [1].

3.2.2 Existující algoritmy

První způsob výpočtu editační vzdálenosti mezi dvěma stromy vznikl v roce 1979. Jednalo se o Taiův algoritmus[3] [58]. Tento algoritmus vznikl na základě

[3]Názvy jsou odvozeny od vědců, kteří přišli na způsob výpočtu.

algoritmu na porovnávání dvou řetězců. Spolu s touto metodou je poskytnut způsob jak převést libovolný řetězec na strom. Metoda spočívá v reprezentaci každého znaku jako uzlu. Všechny nově vytvořené uzly jsou na stejné úrovni a o úroveň výš je přidán virtuální uzel, který je spojen s každým uzlem (znakem). Tato metoda převodu libovolného řetězce na strom není tak užitečná, protože v případě převodu vzniká strom, který neobsahuje větší množství informací o struktuře řetězce než řetězec samotný [58].

Později byl tento výpočet vylepšen Zhang-Shashovým výpočtem vzdálenosti, který snižuje časovou náročnost. Tento algoritmus využívá dynamického programování a strategie mazání určitých nepotřebných uzlů ve stromu. Připadá mu časová složitost $O_{time}(N^2 \cdot M^2)$ [62].

Zhang-Shashův výpočet byl urychlen Kleinovým algoritmem [28]. Na tyto algoritmu navázal Demaine, jeho algoritmus se v současnosti považuje za nejrychlejší techniku výpočtu editační vzdálenosti stromů [13]. Demainův algoritmus má stejnou časovou složitost jako algoritmus Pawlika a Augstena [47].

Zmíněné algoritmy běží s kvadratickou náročností na paměť, tedy přesněji $Ospace(N \cdot M)$. Níže jsou zmíněny časové náročnosti jednotlivých algoritmů.

Taiův algoritmus	$O_{time}(N \cdot M \cdot L_N^2 \cdot L_M^2)$
Zhang-Shashův algoritmus	$O_{time}(N^2 \cdot M^2)$
Kleinův algoritmus	$O_{time}(M^2 \cdot N \cdot log(N)) = O_{time}(N^3 \cdot log(N))$
Demainův algoritmus	$O_{time}(N \cdot M^2(1 + log(N/M))) = O_{time}(N^3)$
Pawlik-Augstenův algoritmus	$O_{time}(N \cdot M^2(1 + log(N/M))) = O_{time}(N^3)$

Tabulka 3.1: Tabulka složitostí výpočtu editační vzdálenosti stromů

3.2.3 Výpočet pomocí dynamického programování

Výpočet pomocí dynamického programování je nejpomalejší ze zmíněných přístupů, avšak pro účel pochopení výpočtu slouží nejlépe, protože neobsahuje optimalizace, které by mohly ztížit pochopeni. Níže je algoritmus popsán.

- Převedení stromu do post-order[4] průchodu zleva doprava.
- T[i] značí uzel na pozici i.

[4]Více informací o průchodu stromem lze nalézt zde https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/

- l(i) značí nejlevějšího potomka (list) stromu s kořenem na T[i], V případě že T[i] je list, poté l(i) je i.
- r(i) značí nejpravějšího potomka (list) stromu s kořenem na T[i]. V případě že T[i] je list, poté r(i) je i.
- p(i) značí rodiče uzlu T[i].
- desc(i) značí potomky uzlu T[i].
- Q[i..i', j..j'] značí editační vzdálenost mezi T[i..i'] a T[j..j'] (více informací v kapitole 3.1).

Poté nalezení editační vzdálenosti pro jakékoliv i $\in$ desc(i_1) a j $\in$ desc(j_1), tedy $Q[l(i_1)..i, l(j_1)..j]$ je rovno minimální hodnotě z následujících hodnot [61]:

- 1) $Q[l(i_1)..i-1, l(j_1)..j] + p(i, \perp)$,
- 2) $Q[l(i_1)..i, l(j_1)..j-1] + p(\perp, j)$,
- 3) $Q[l(i_1)..l(i)-1, l(j_1)..l(j)-1] + Q[l(i)..i-1, l(j)..j-1] + p(i, j)$.

$p(i, \perp)$ značí, že i bylo smazáno (první rekurze), $p(\perp, j)$ značí, že j bylo smazáno (druhá rekurze). Ve třetím případě je $T_1[l(i_1)..l(i)-1]$ namapováno do $T_2[l(j_1)..l(j)-1]$ a $T_1[l(i)..i-1]$ je namapováno do $T_2[l(j)..j-1]$ [61].

3.2.4 Programy pro výpočet editační vzdálenosti

V dnešní době existuje mnoho implementací těchto algoritmů nad sekundárními RNA strukturami. Mezi tyto nástroje patří například SimTree[5]. Jedná se o webový a desktopový nástroj implementovaný v jazyce Java. Program očekává na vstupu dvě RNA struktury a jeho výstupem je skóre podobnosti a mapování mezi strukturami, které lze použit pro identifikování podobných podstruktur [16].

Nástroj ViennaRNA[6] obsahuje mnoho podprogramů. Mimo porovnávání více sekundárních struktur pomocí editace stromů (podprogram RNAdistance [53], RNAforester [27]) obsahuje také jiné nástroje pro práci s RNA, například nástroje pro predikci sekundární struktury ze struktury primární [25] [33].

[5] http://bioinfo.cs.technion.ac.il/SimTree/
[6] https://www.tbi.univie.ac.at/RNA/

3.3 Ostatní přístupy výpočtu podobnosti

Tato kapitola popisuje přístupy k výpočtu podobnosti RNA sekundárních struktur, které se nedalo zařadit do žádné z předcházejících sekcí.

Sekundární RNA struktury lze porovnávat také pomocí aplikace kódovacího algoritmu Lempel-Ziv(LZ) na sekundární strukturu. LZ algoritmus je založený na extrakci opakujících se motivů z textu. Při aplikaci na sekundární strukturu dokáže algoritmus extrahovat informaci o opakujících se vzorech ve struktuře. Tato metoda dosahuje časové složitosti $O_{time}(N^2)$ a paměťové složitosti $O_{space}(N^2)$ [31].

Hlavním problémem této metody je, že při použití LZ algoritmu je potřeba převést nelineární RNA sekundární strukturu na lineární sekvence charakteristik. V tomto kroku se může ztratit část informace o sekundární struktuře RNA [31].

3.4 Shrnutí současných metod

V současné době existuje mnoho přístupů k porovnávání sekundárních RNA struktur. Mezi triviálnější metody patří metody založené na porovnávání řetězců, které dosahují relativně dobré časové náročnosti. Tyto metody jsou v dnešní době avšak užívané hlavně pro porovnávání primárních struktur RNA [56].

K porovnávání sekundárních struktur RNA jsou v dnešní době často užívané metody fungující na základě porovnání editačních vzdáleností mezi dvěma stromy. Tyto metody poskytují často velmi dobré výsledky, avšak jejích časová složitost je horší, než například u metod porovnávající pouze řetězce [13]. Existují i alternativní metody, které zpravidla zanedbávají část informace ze sekundární struktury výměnou za vylepšení časové složitosti výpočtu [31]. Shrnutí časových složitosti vybraných algoritmů lze vidět v tabulce 3.2.

Výpočet editační vzdálenosti stromů	$O_{time}(N^3)$ [13]
Levenshteinova vzdálenost	$O_{time}(N^2)$ [30]
Výpočet založený na Lempel-Ziv metodě	$O_{time}(N^2)$ [31]

Tabulka 3.2: Tabulka složitostí algoritmů porovnávajících sekundární struktury RNA

Kapitola 4

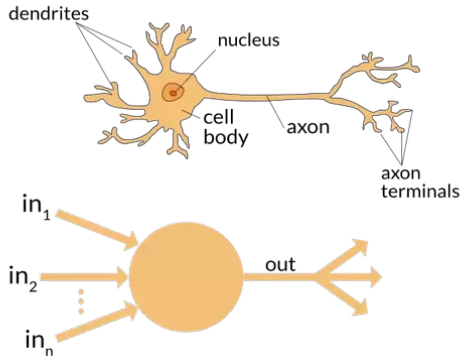
Metody strojového učení

Tato kapitola popisuje metody strojového učení, které jsou využívány v nové vytvořeném algoritmu Struc2Vec (více o Struc2Vec v kapitole 5). Na začátku této kapitoly je popis neuronových sítí a detailněji jsou popsány metody Word2vec a Doc2vec. Metoda Doc2vec bude základem návrhu metody Struc2Vec, pouze s tím rozdílem, že namísto dokumentů složených ze slov budeme využívat sekundární struktury RNA. Pro posuzování kvality výsledků dosažených pomocí Struc2Vec metody budou využity metody redukce dimenze a metoda podpůrných vektorů. Tyto metody jsou také popsány v této kapitole.

4.1 Neuronové sítě

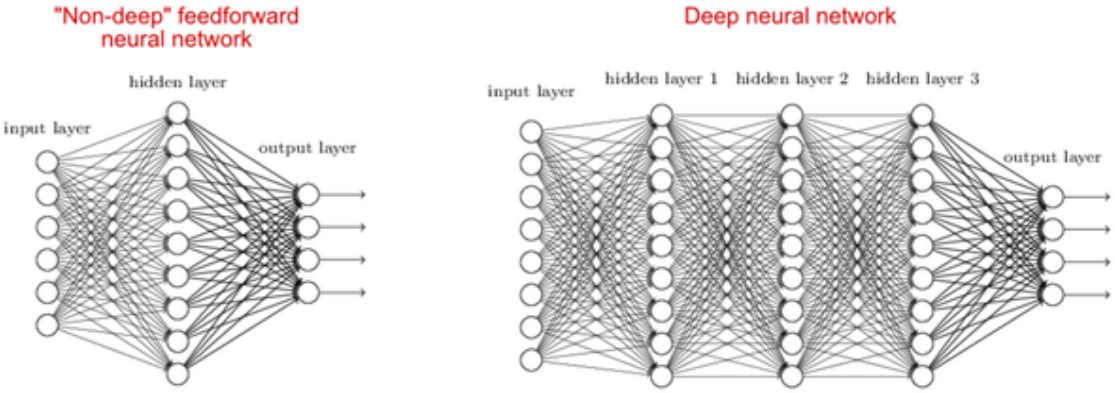
Neuronové sítě (ANN, z angl. Artificial Neural Network) jsou abstraktním matematickým modelem, který je inspirován mozkem, jeho funkcemi a mechanismy, které v něm pracují [19]. Tyto sítě lze definovat více způsoby, jedna z definic je, že se jedná o výpočetní systém tvořený z mnoha jednoduchých, vzájemně propojených elementů. které zpracovávají informace na základě svého dynamického stavu, který lze ovlivnit pomocí externího vstupu [8].

Jedním z prvních modelů připomínajících lidský mozek byl perceptron. Jedná se o algoritmus sloužící k učení binárních klasifikátorů. Výsledkem implementace perceptronu je funkce rozhodující, zdali vstup (vektor čísel) patří do určité třídy [3] [51].



Obrázek 4.1: Vizualizace rozdílu mezi biologickým a umělým neuronem [2].

Neuronové sítě jsou utvořeny z mnoha jednoduchých procesorů připomínajících perceptrony, které se v praxi nazývají neurony. Z těchto procesorů jsou utvořeny vrstvy. První vrstva se nazývá vstupní vrstva, poslední vrstva se nazývá výstupní vrstva. Všechny vrstvy mezi těmito dvěma vrstvami se nazývají skryté. Pokud neuronová síť obsahuje pouze jednu skrytou vrstvu, jedná se o mělkou neuronovou sít, v opačném případě jde o hlubokou neuronovou síť (lze vidět na obrázku 4.2) [52].



Obrázek 4.2: Rozdíl mezi mělkou a hlubokou neuronovou sítí [18].

Jednotlivé neurony, obsahují určitou aktivační funkci, která reaguje na signál z předešlé vrstvy a vysílá signál do následující vrstvy (signálem je v tomto případě číslo, často desetinné číslo z intervalu <0,1>).

Učení neuronových sítí lze rozdělit na učení s učitelem a učení bez učitele. Učení s učitelem znamená, že neuronovou síť je potřeba nejprve naučit na trénovacích datech, o kterých známe odezvu, kterou chceme aby nám neuronová síť vrátila (například třídu v případě klasifikace). V trénovací fázi jsou jednotlivé trénovací vzorky předkládány neuronové síti a podle odezvy jsou nastavovány parametry jednotlivých neuronů takovým způsobem,

aby odezva co nejvíce odpovídala realitě [52]. Trénování je často prováděno iterativně. Existuje více přístupů pro jak zjistit, že je vhodné ukončit adaptaci neuronové sítě. Mezi nejčastější typy trénování patří například iterování, dokud není dosažené určité chyby na trénovacích datech a nebo přímo zvolený počet iterací. U učení bez učitele není předem známá realita o trénovacích vzorcích, protože neuronová síť si sama vytváří skupiny, do kterých dělí vstupy. Pro rozdělování do skupin je možné využít například metodu k-means[1] [48].

Využití neuronových sítí lze nalézt v napodobování biologických nervových systémů a inteligence, jako adaptivní procesor nebo ovladač implementovaný v hardwaru pro aplikace jako jsou například roboti. Další využití lze nalézt například v metodách pro analýzu dat nebo v počítačovém vidění [48][52].

4.2 Word2Vec

Tato kapitola popisuje metodu Word2vec, která je založena na principu mělkých neuronových sítí.

Pro samotný proces trénování neuronové sítě Word2vec je potřeba korpusu. Korpus lze neformálně chápat jako soubor různých textů (knihy apod.), ve kterých je možné vyhledávat jazykové jevy (slovní spojení, kontext...) [11]. Pro účely Word2vec jsou jazykové jevy v korpusu velmi důležité, jelikož na kontextu slov záleží při trénování algoritmu. Vzhledem k tomu, že je na trénování použit pouze trénovací korpus slov, může dojít k nevhodnému naučení neuronové sítě v případě špatného (např. velmi rozdílného od testovacích dat) zvolení trénovacího korpusu. V takovém případě nemusí Word2vec správně odhadovat vztahy mezi jednotlivými slovy na testovacích datech [50]. Například pokud v trénovacím korpusu nebude reprezentován vztah mezi nocí a tmou, nebude mít algoritmus možnost se tuto spojitost naučit.

Při trénování modelu na korpusu je vytvořena určitá slovní zásoba modelu (z angl. vocabulary). Slova, která se nacházela v korpusu jsou uložena v této slovní zásobě. V trénovací fázi lze určitá slova združit do jednoho a nebo ignorovat slova, která se nachází v trénovacích datech velmi zřídka [50].

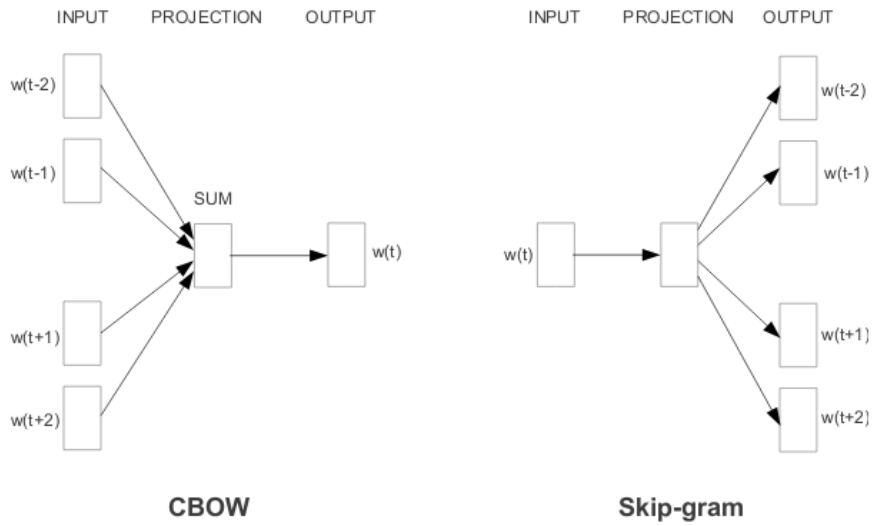
Existují dva typy modelů metody Word2vec – CBOW (z angl. Continuous Bag Of Words) a skip-gram. Obě tyto metody vytváří pro každé slovo dva vektory – vstupní a výstupní. Učení vstupních vektorů není časově náročné,

[1] https://cs.wikipedia.org/wiki/K-means

ale učení výstupních vektorů je časově velmi náročné. Pro naučení výstupních vektorů je potřeba iterovat přes všechny slova z naučené slovní zásoby a provádění této iterace je časově velmi náročné. Tento problém se ve Word2vec často řeší pomocí hierarchického softmaxu (jedná se o efektivní výpočet funkce softmax[2] pomocí reprezentace slovní zásoby jako binárního stromu) [50].

Rozdíl mezi CBOW a skip-gramem je v typu vstupu a výstupu. Při CBOW jsou vstupem váhy slov v kontextu daného slova a výstupem je váha daného slova. Algoritmus skip-gram funguje přesně naopak, tedy vstupem je váha daného slova a výstupem jsou váhy slov v kontextu [36]. Velikost kontextu lze ovlivnit pomocí velikosti kontextového okna. Algoritmus skip-gram poskytuje lepší výsledky v případě slov, které se v trénovacím korpusu nacházejí jen velmi zřídka. Naopak při u slov nacházejících se v korpusu často poskytuje lepší výsledky metoda CBOW. Díky těmto vlastnostem je vhodné jej využívat v případě nízkého počtu trénovacích dat [36]. Rozdíl lze vidět na obrázku 4.3.

Časová náročnost učení Word2vec algoritmu je $\mathcal{O}_{time} = E \cdot T \cdot Q$, kde E je počet epoch učení, T je počet slov v trénovacím korpusu a Q je definováno pro skip-gram a CBOW rozdílně. Pro skip-gram je Q definováno jako $Q = C \cdot (D + D \cdot log_2(V))$, kde C je maximální vzdálenost slov, tedy pokud zvolíme C=5, poté je velikost kontextu náhodným číslem z intervalu $< 1, 5 >$. D je počet neuronů ve skrýté vrstvě a V je velikost korpusu, zatímco pro CBOW je Q definováno $Q = N \cdot D + D \cdot log_2(V))$, kde N znamená počet aktivních vstupních neuronů. Z těchto informací lze usoudit, že trénování pomocí metody CBOW je rychlejší na naučení [29].



Obrázek 4.3: Vizualizace rozdílu mezi CBOW a skip-gram [36].

[2]https://en.wikipedia.org/wiki/Softmax_function

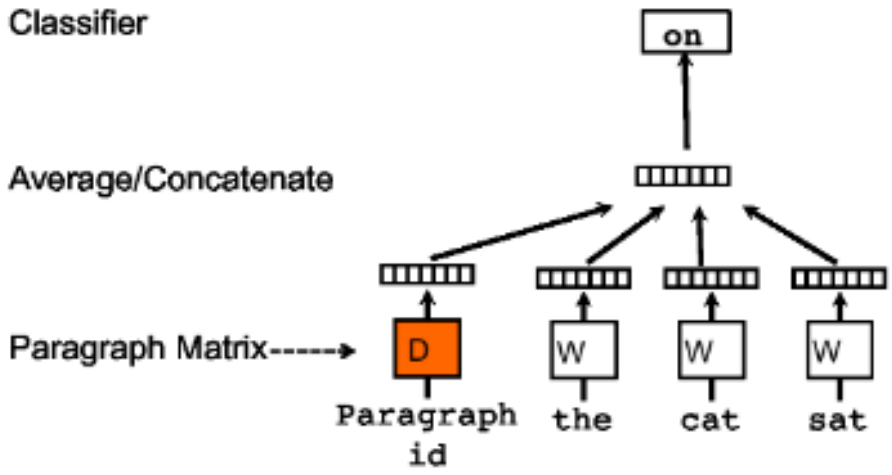
Ačkoliv lze vzdálenosti mezi vektory vytvořenými pomocí Word2vec porovnávat různými způsoby, zmiňuje autor tohoto algoritmu použití kosinové podobnosti[3] (z angl. cosine similarity) [36]. Jelikož tato metoda bere při porovnávání velikost vektorů a úhel mezi nimi, může při porovnání podobnosti dosahovat výsledná podobnost i záporných hodnot.

4.2.1 Doc2Vec

Klasifikace textových dokumentů (vět, paragrafů apod.) je hojně využívána v různých odvětvích počítačových věd, například ve webovém prohledávání, filtrování spamu nebo vyhledávání dokumentů. Algoritmy řešící tuto úlohu zpravidla potřebují reprezentovat vstupní text jako vektor s fixní velikostí [29]. Mezi algoritmy řešící tento problém patří v dnešní době například bag-of-words[4], který využívá četnosti daných slov, ale ignoruje ostatní vlastnosti textu, jako je například kontext. Ztráta kontextu a pořadí slov může způsobit, že dvě věty budou mít velmi podobný vektor, ačkoliv budou velmi rozdílné [22]. Největší slabinou metody bag-of-words je tedy zanedbání sémantiky textu. Tato kapitola popisuje metodu Doc2vec, která funguje na základě popisu celých vět (paragrafů etc.) pomocí vektoru, zatímco slova korpusu jsou naučena zvlášť. Vektory slov i vět jsou učeny pomocí technik klesání pomocí gradientu[5] a pomocí zpětné propagace[6]. Tato metoda je silně založena na principu Word2vec a využívá velmi podobný princip na tvorbu vektorů jednotlivých slov [29].

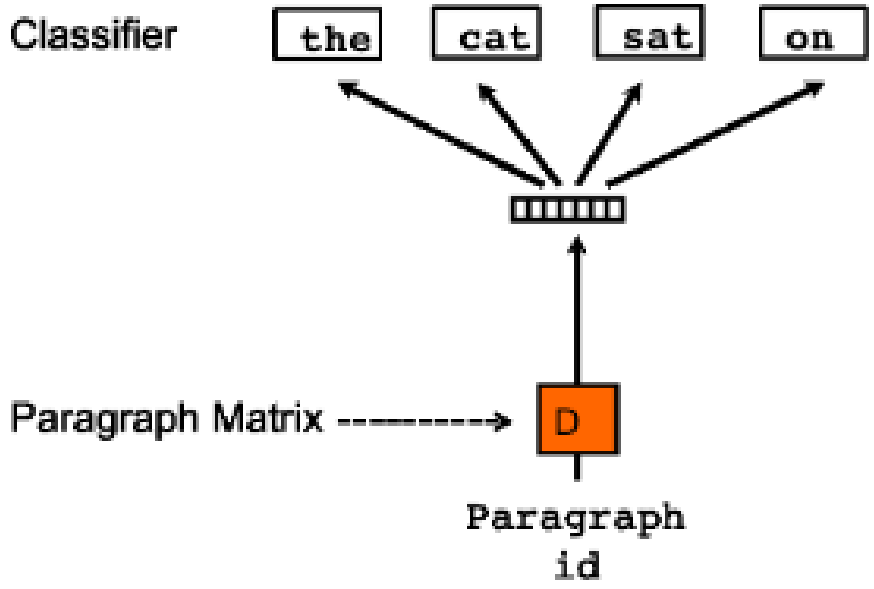
Trénování vektorů vět je v metodě Doc2vec založeno na metodách pro trénování vektorů slov z Word2vec. První z metod používaných při trénování algoritmu je distribuovaný paměťový model paragrafových vektorů (PV-DM, z angl Distributed Memory Model of Paragraph Vectors). V této metodě je každá věta mapována do vektoru o stejné velikosti, jako jsou vektory jednotlivých slov. Narozdíl od vytváření vektorů slov, je v při vytváření vektorů vět použit i vektor věty, což je rozdílné od Word2vec. Tento princip lze je vizualizován na obrázku 4.4. Při predikci je potřeba provést inferenční krok, ve kterém se počítá vektor pro nový paragraf. Tento inferenční krok se také počítá pomocí metody klesání pomocí gradientu – při odhadování vektoru věty je tedy použit iterativní přístup a není zaručeno, že dva odhady stejného dokumentu budou mít stejný výsledek. metoda PV-DM poskytuje většinou lepší výsledky, než metoda PV-DBOW (viz níže) [29].

[3] https://en.wikipedia.org/wiki/Cosine_similarity
[4] https://en.wikipedia.org/wiki/Bag-of-words_model
[5] https://en.wikipedia.org/wiki/Stochastic_gradient_descent
[6] https://en.wikipedia.org/wiki/Backpropagation



Obrázek 4.4: Vizualizace trénování vektoru v metodě Doc2vec – PV-DM [29].

Další technikou Doc2vec distribuovaný batoh slov paragrafového vektoru (PV-DBOW, z angl. Distributed Bag of Words of Paragraph Vector). Tato metoda zanedbává kontext slov ve vstupu. Namísto toho předpovídá slova náhodně zvolená z výstupu. Toto znamená, že při každé iteraci klesání pomocí gradientu je vzorkováno okno z textu o určité velikosti a poté je provedena klasifikace na základě vektoru věty. Tato technika je paměťově méně náročná než technika PV-DM. PV-DBOW je velmi podobná technice skip-gramu z Word2vec [29]. Tato technika je vizualizována na obrázku 4.5.



Obrázek 4.5: Vizualizace trénování vektoru v metodě Doc2vec – PV-DBOW [29].

Po natrénování Doc2vec modelu lze odhadnout i vektor dokumentu, který nebyl přítomen při trénování. Dokument, který odhadujeme by se měl skládat z již známých slov, tedy jednotlivá slova v novém dokumentu by měla být přítomna i někde v trénovacích datech, pokud tato podmínka nebude splněna, budou slova algoritmem ignorována [29].

Parametry Doc2vec

Tato kapitola popisuje některé konfigurovatelné parametry metody Doc2vec. Parametry popsané v této kapitole jsou dostupné v knihovně Gensim [63], která implementuje metodu Doc2vec.

- dm – určuje trénovací algoritmus. Pokud je hodnota dm 1, je využit PV-DM typ algoritmu, jinak je využit PV-DBOW
- vector_size – počet dimenzí vektorů natrénovaného modelu
- alpha – určuje počáteční rychlost učení
- min_alpha – alpha se lineárně zmenšuje, dokud nedosáhne této hodnoty
- min_count – slovo se musí objevit v testovacích datech alespoň min_count násobně, jinak bude vynecháno
- epochs – určuje počet iterací při trénování modelu. Větší množství epoch znamená větší množství průchodů přes trénovací data
- steps – Označuje počet iterací pro vnoření nového dokumentu do natrénovaného prostoru. Vyšší hodnoty znamenají delší dobu vnořování, ale zlepšují kvalitu a stabilitu vektoru dokumentu [29]

4.3 Metoda podpůrných vektorů

Metoda podpůrných vektorů (z angl. support vector machine, SVM) je metoda strojového učení s učitelem sloužící pro řešení dvouskupinových klasifikačních problémů. Koncept této metody je založen na tvorbě lineárního n-1 dimenzionálního prostoru(nadroviny) ve n-dimenzionálním prostoru, který prostor rozdělí na dvě části. Pokud je n-dimenzionální prostor lineárně oddělitelný, poté v ideálním případě reprezentuje každá ze dvou částí prostoru jednu

klasifikační třídu [10]. Tato metoda se bude ve vytvořené metode Struc2Vec využívat jako jedna z možností pro klasifikaci sekundárních RNA struktur.

Pro tvorbu nadroviny je potřeba mít trénovací data v následujícím tvaru:

$$(y_1, x_1), ..., (y_t, x_t), y_i \in \{-1, 1\}$$

Aby byl prostor lineárně oddělitelný, je potřeba aby existoval vektor w a skalár b, pro které plati:

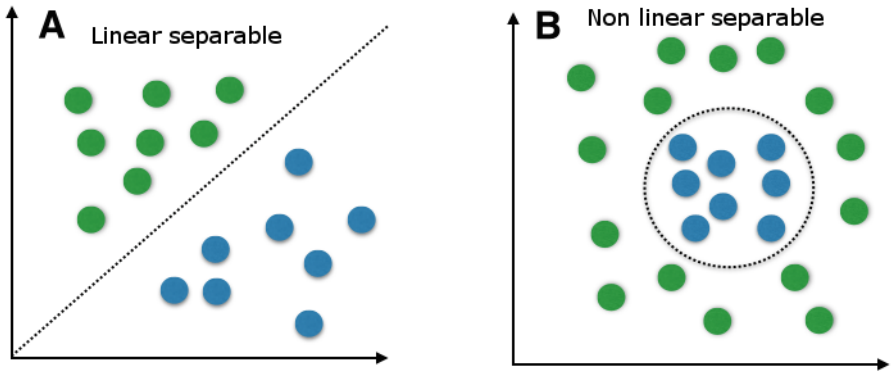
$$w \cdot x_i + b \geq 1 if y_i = 1$$
$$w \cdot x_i + b \leq 1 if y_i = -1$$

Poté je lze definovat nadrovinu, která rozděluje ideálně rozděluje prostor jako:

$$w_0 \cdot x + b_0 = 0$$

Ideální nadrovina je taková, která rozděluje prostor s maximální vzdáleností od bodů jednotlivých tříd [10].

Tato metoda lze použít pro rozdělení prostoru vytvořeného pomocí metody Doc2vec v případě, že vytvořený prostor bude lineárně oddělitelný. Příklad lineárně oddělitelného a lineárně neoddělitelného prostoru lze vidět na obrázku 4.6, na levém obrázku lze vidět jak SVM (tečkovaná čára) odděluje prostor na dvě části.



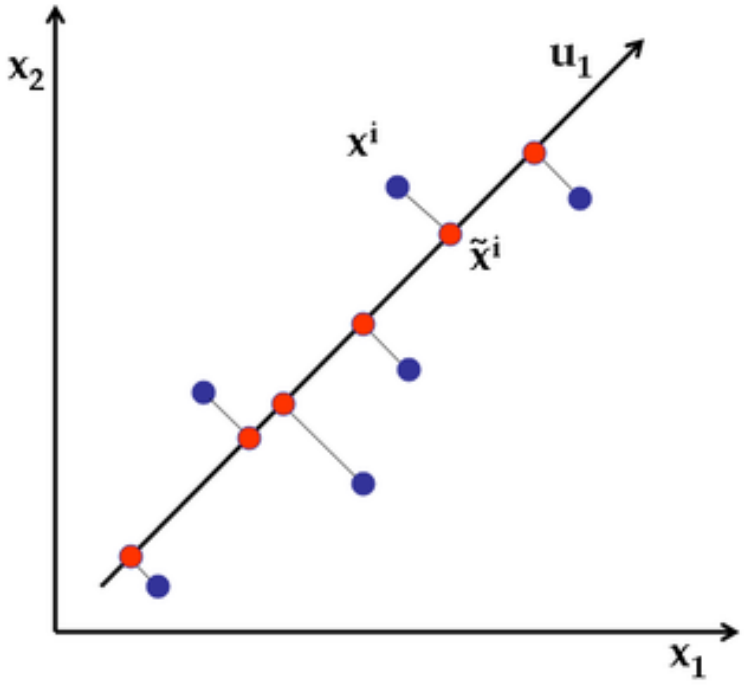
Obrázek 4.6: Lineární oddělitelnost prostoru [49].

4.4 Analýza hlavních komponent

Pro vizualizaci prostoru, ve kterém se vektory vytvořené pomocí Word2vec (případně Doc2vec) nacházejí, je potřeba snížit počet dimenzí vytvořeného prostoru. Tato kapitola popisuje jednu z užívaných metod sloužících k redukci dimenze – analýzu hlavních komponent (z angl. principal component analysis, PCA).

Cílem této metody je extrakce důležitých informací z vice-dimenzionalního prostoru a reprezentovat jej pomocí nových ortogonálních (kolmých) proměnných, které se nazývají hlavní komponenty. Mimo jiné pomáha PCA k zmenšení velikosti dat zachováním pouze důležitých informací a také zjednodušuje popis data setu [1].

Hlavní komponenty jsou získány z lineární kombinace originálních proměnných prostoru. První hlavní komponenta má největší možný rozptyl (tím zachová co nejvíce informací). Druhá hlavní komponenta musí být ortogonální k první komponentě a zároveň musí mít co největší rozptyl. V případě, že redukujeme do více než dvou dimenzionálního prostoru, jsou další hlavní komponenty vytvořeny stejným způsobem, jako komponenta druhá. Touto metodou lze tedy redukovat dimenze n-dimenzionálního prostoru na libovolně velký m-dimenzionální prostor za předpokladu, že $m < n$.. Tato metoda nemusí vždy vracet ideální výsledky, jelikož největší rozptyl nemusí nutně znamenat nejvíce informací [1]. Příklad vizualizace redukce dimenze z 2D na 1D lze vidět na obrázku 4.7, kde modré body značí původní body v 2D prostoru a červené body značí body namapované do 1D prostoru.



Obrázek 4.7: Vizualizace redukce dimenze z 2D na 1D pomocí PCA [39].

Část II

Praktická část

Kapitola 5

Struc2Vec

Tato kapitola popisuje nově vytvořený algoritmus Struc2Vec (S2V), který je podobně jako Doc2Vec založen na principu vnořování slov a mělkých neuronových sítí. Cílem algoritmu Struc2Vec je výpočet podobnosti sekundárních RNA struktur, kde rychlost výpočtu je výrazně nižší než u výše popsaných metod. Tato kapitola popisuje především postup při vytváření slov reprezentujících sekundární strukturu. Tvorba slov je velmi důležitou částí algoritmu, nevhodná volba slov může zapříčinit nesprávná zanoření. Tvorba slov také algoritmus Struc2Vec odlišuje od dříve popsané metody Doc2Vec. Doc2Vec tvoří vektory reprezentující textové dokumenty, Struc2Vec tvoří vektory reprezentující jednotlivé sekundární RNA struktury. V případě zanořování dokumentů není definice slov nutná, protože slova dokument přímo tvoří. Chceme-li zanořovat sekvence je třeba jejich stavební bloky analogické slovům nejprve definovat.

5.1 Tvorba slov

Tato kapitola popisuje postup při vytváření slov, které jsou dále v algoritmu používané stejně, jako jsou používány slova u Doc2vec. Cílem při vytváření slov je co nejpřesnější definice sekundárních struktur.

5.1.1 Pomocí vzdáleností a délek

Tato metoda bere v potaz pouze sekundární strukturu RNA, ignoruje tedy její primární strukturu. Domníváme se, že tato vlastnost je pozitivní, jelikož například IRES hepatitidy C, který bude předmětem návazného zkoumání, mění svou primární strukturu tak silně, že jej není podle ní možné rozpoznat. Jednotlivá slova se vytváří tím způsobem, že zjistíme vlastnosti části RNA (část je například vlásenka apod.) a ty transformujeme na slova.

Část sekundární struktury RNA je definována tak, že báze patří do současné struktury pokud nesplňuje ani jednu z níže uvedených podmínek. V případě, že některá z podmínek je splněna, patří báze již do další části RNA.

- předchozí báze byla spojená s jinou bází, ale současná báze není spojená s jinou bází (v závorkovém formátu jde o následující přechod: (**(**...)))
- předchozí báze nebyla spojená s jinou bází, ale současná báze je spojená s jinou bází (v závorkovém formátu jde o následující přechod: ((...**)**))
- předchozí báze byla spojená s jinou bází, současná báze je také spojená s jinou bází, ale tyto báze nejsou spojeny se stejnou částí struktury RNA. (v závorkovém formátu jde o následující přechod: (**(**(..)).))

U jednotlivých částí se získávají následující vlastnosti:

- délka současné části RNA
- vzdálenost napojení na druhou část RNA (V případě, že existuje). Bere se v potaz vždy nejkratší vzdálenost mezi jednotlivými částmi. Vzdálenost může být i záporná (pokud je popisována báze ve struktuře dále, než báze se kterou je spojená)

Kombinací těchto dvou vlastností jsou utvořena slova, která jsou nadále používána při výpočtu podobnosti. Při tvorbě slov je dbáno na to, aby všechna slova byla unikátní, tedy aby kombinaci různých vlastností nevznikla dvě stejná slova. Unikátnosti je dosaženo pomocí vložení znaku "x"mezi jednotlivé vlastnosti struktury. Výsledná slova na příkladu ((...))..() budou složena z následujicích vlastností:

2, 4 – Počet prvních otevíracích závorek je 2. Vzdálenost mezi prvními závorkami a příslušnými ukončovacími závorkami je 4.

3, 0 – Počet následujících nespárovaných nukleových bází je 3 (naznačeno tečkami v zápise). Vzhledem k tomu, že tyto báze nejsou spárovány, je vzdálenost propojení 0.

2, -4 – Stejné jako u prvních závorek, vzdálenost je negativní, jelikož spárované báze jsou nalevo od současné pozice ve struktuře. Negativní vzdálenost nastává pokaždé, když je závorka uzavřena.

2, 0 – Dvě nespojené báze.

1, 1 – Jedna báze, která je spojena s bázi ve vzdálenosti jedna.

1, -1 – Jedna báze, která je spojena s předešlou bází.

Tato metoda bude našim základním kamenem při tvorbě slov pro metodu Struc2Vec. Všechny výsledky, grafy apod. uvedené v kapitolách níže byly dosaženy za použití této metody tvorby slov, pokud není uvedeno jinak.

Redukce počtu slov

Vzhledem k tomu, že vzdálenost i délka jsou číselné hodnoty, lze je rozdělit do "binů"(z angl. data binning[1]) podle jedné nebo obou z těchto hodnot. Cílem této techniky je dosáhnutí výrazně menšího počtu slov využívaného v S2V. Tato metoda stojí za vyzkoušení v případě, že trénovací set dat je příliš malý a S2V pomocí standardní metody nedává dobré výsledky.

Při "binování"dle vzdálenosti je potřeba zachovat informaci, že báze je spojená s jinou i v případě, že hodnota vzdálenosti je tak malá, že spadá do "binu"kam spadá i nespojená báze. Jak je tohoto docíleno je vidět na příkladu níže.

Výsledná slova na příkladu ((...))..() s "binováním"dle délky 2 a dle vzdálenosti 3 budou vypadat následovně:

2, 3 – Délka 2 zůstává beze změny, avšak vzdálenost 4 je pozměněna na nejbližší spodní "bin"– tedy 3.

[1] https://en.wikipedia.org/wiki/Data_binning

2, 0 – Počet následujících nespárovaných nukleových bází je 3, avšak je nutno zaokrouhlit na spodní hranici "binování"podle délky

2, -3 – Stejně jako u prvních závorek je potřeba změnit vzdálenost na nejbližší "bin"– tedy -3

2, 0 – Dvě nespojené báze.

0, 1 – Jedná báze je změněna na počet 0, vzdálenost 1 je zachována, aby bylo možné rozpoznat spojené a nespojené báze.

0, -1 – Opět zachováváme informaci o spojených bázích.

Výchozí nastavení S2V metody je bez redukce počtu slov, protože u velkého množství testovacích dat může tato redukce uškodit kvalitě výsledku. Avšak pokud při spouštění aplikace nemáte dostatečné množství dat, je možné vyzkoušet i možnost binování, v určitých případech dosahuje lepších výsledků (více informací v sekci 8.3).

Pro anotaci metody Struc2Vec s redukcí počtu slov budeme v dalších sekcích užívat *Struc2Vec (x,y)*, kde x značí redukci slov podle délky a y značí redukci slov podle vzdálenosti.

5.1.2 Sousední slova

Tato metoda bere v potaz jak primární, tak sekundární strukturu. Jednotlivá slova jsou vytvořena pomocí sousedních bází. Na každé straně báze je jeden soused a pokud je báze propojená s jinou bází, bere se v potaz i tato báze. U všech bázi s výjimkou krajních existují vždy alespoň dvě sousední báze (u krajních může existovat pouze jedna sousední báze).

Slova jsou vždy složena ze čtyř znaků (z abecedy o znacích ACGU-). První znak znázorňuje aktuální bázi. Druhý znak znázorňuje levého souseda (předešlý znak). Třetí znak spojenou bázi (v případě znaku otevření závorky u sekundární struktury, se jedná o bázi na pozici příslušného ukončení této závorky v sekundární struktuře). Čtvrtý znak znázorňuje pravého souseda (následující znak). Pokud kterákoliv ze čtyř bází chybí, je báze nahrazena znakem "-".

Výsledná slova na příkladu primární struktury GGAACC o sekundární struktuře ((..)) budou tedy vypadat následovně:

- G-CG – První báze je G, levý soused chybí, jelikož se jedná o první bázi ve struktuře.
- GGCG – Druhá báze má všechny sousedy.
- AG-A – Třetí báze nemá spojenou bázi, proto je na třetí pozici znak "-".
- AA-C – Obdobný případ jako u třetí báze.
- CAGC – Obdobný případ jako u druhé báze.
- CCG- – Poslední báze v příkladu je C, chybí pravý soused.

Tato metoda má několik problémů při použití zároveň s metodou S2V. První problém je malé množství vygenerovaných slov. Každá báze může mít na své straně 4 možné báze, nebo chybějící bázi (kromě báze, kterou zrovna určujeme). Množství vygenerovaných slov by tedy bylo pouze $5^3 \cdot 4$, tedy 500. Pokud ale vezmeme v potaz, že pouze první a poslední báze ve struktuře nemá levého (respektive pravého) souseda, zmenšuje se počet slov na $4^3 \cdot 5$, tedy 320. Protější (spojená) báze může sice teoreticky nabývat všech hodnot, ale spojení probíhají s velkou pravděpodobností pouze mezi bázemi guaninu a cytosinu, případně uracylu a adeninu. Pokud bychom tedy zanedbali málo vyskytované vytvořené slova, byl by počet slov roven $4^3 \cdot 3$, což je pouze 192 slov.

Dalším problémem metody je tvoření kontextu pomocí tvorby slov, nikoliv pomocí samotného algoritmu (Doc2vec). Metoda S2V je vhodná právě kvůli samotné tvorbě kontextu a tvořit kontext na úrovni slov by mohlo být zbytečné.

Posledním zmíněným problémem je, že metoda nedokáže zachytit určité strukturní motivy. Popisuje vždy pouze současnou bázi. Řešení tohoto problému by mohlo spočívat ve zvětšení kontextového okénka u algoritmu S2V.

Kapitola 6

Data

Tato kapitola popisuje data použitá pro implementaci, experimenty a otestování metody S2V. První sekce popisuje data stažená z veřejně dostupných databází. Druhá sekce popisuje uměle vytvořená data.

Data, která je možné použít pro naučení modelu, musí být ve formátu *bpseq*, *db* nebo *ct*.

Při porovnávání sekundárních RNA struktur se doporučuje vytvořit si model se svými daty, o kterých má uživatel nějaké informace (domněnka, že existují strukturní rozdíly, účel pozitivních/negativních příkladů apod.). Více informací o spouštění programu je popsáno v kapitole 7.

6.1 Reálná data

K testovacím účelům byla využita data z RNA databáze RNase P sekundárních struktur[1] [6] a také Comparative Rna Website (CRW)[2] [7].

Data stažená z výše zmíněných databází sloužila k otestování algoritmu S2V na reálných datech, na experimentální zjištění vhodných parametrů pro

[1] http://www.mbio.ncsu.edu/RNaseP/home.html
[2] http://www.rna.icmb.utexas.edu/

algoritmus a také pro tvorbu příkladu, který si může vyzkoušet každý, kdo si stáhne kód této diplomové práce. Detail těchto dat lze vidět v tabulce 6.1.

Při stažení kódu se ve složce *data* nachází shell skript *downloadStructures.sh*, který slouží ke stažení deseti typů sekundárních RNA struktur[3], které lze využít k natrénování modelu.

Data, která se nevyužívají při tvorbě modelu se nacházejí ve složce *data-not-used*. Důvodem, proč se některá data nepoužívají, je například nízké zastoupení určité třídy struktur. Pokud je za třídu k dispozici pouze jedna struktura, model nemusí být pro danou třídu reprezentativní. V této práci jsme se omezili na testy a předpovědi taxonomických kategorií především pro dobře zastoupené třídy struktur. Vycházíme přitom ze zjednodušené hypotézy příbuznosti mezi taxonomickou třídou sekvence a její sekundární strukturou. Tato hypotéza byla potvrzena i experimentálně za použití tradičních metod pro porovnání sekundárních struktur RNA.

Název třídy	Počet	Původ	Průměr	Max	Min
alpha purple bacteria	22	RNase P	358	480	290
beta purple bacteria	14	RNase P	306	415	238
chlamidia	11	RNase P	356	250	409
crenearcheaeal	8	RNase P	290	331	260
euryarchaeal	31	RNase P	313	476	230
gammaPurupleBacteria	87	RNase P	316	400	256
greenNonSulfurBacteria	6	RNase P	326	351	288
nuclear	56	RNase P	308	466	190
planctomycetes	9	RNase P	360	394	321
spirochaete	7	RNase P	333	353	286
bacteria	410	CRW	1606	3126	111
eukaryota	338	CRW	1146	3543	122

Tabulka 6.1: Tabulka použitých reálných dat

6.2 Uměle vytvořená data

Pro testovací účely byla také využita data, která byla manuálně vytvořena, ať už vygenerována pomocí nástrojů na vytvoření sekundární struktury z primární[4] nebo vytvořena ručně.

[3]Tento skript je funkční ke květnu 2019, jelikož záleží na externí webové stránce, muže se stát, že v budoucnu funkční nebude. Data ale lze stáhnout i manuálně ze zmíněných databází.

[4]Byl použit nástroj, který lze najít zde: http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi

Vygenerované sekundární struktury byly vytvořeny z náhodných primárních struktur (náhodný řetězec z písmen "A", "G", "U"a "C"). Účelem těchto struktur je porovnání rychlosti S2V s jinými přístupy porovnání sekundárních RNA struktur. Tato data lze nalézt ve složce *data-test/benchmark*. Podsložky v této složce jsou pojmenovány podle délky sekundárních RNA struktur, které byly použité pro zjištění rychlosti algoritmu, více informací v kapitole 8.1.

Ručně vytvořená data sloužila pro původní testy přesnosti algoritmu, tedy zdali je algoritmus schopný poznat, že struktury, které se liší jen velmi minimálně budou algoritmem posouzeny jako velmi podobné. Příkladem těchto dat je například prodloužení vyboulené smyčky o jednu bázi apod. Tato data se nacházejí ve složce *data-test/loops*.

Kapitola 7

Implementace

Tato kapitola popisuje implementaci algoritmu S2V, strukturu kódu, parametry kódu, vstupní soubory a ostatní detaily, které mohou být velmi užitečné při používání nebo rozšiřování kódu.

Struc2Vec algoritmus je naimplementován v jazyce Python ve verzi 3.5. Programovací prostředí použité při implementaci je PyCharm[1]. Kód také obsahuje několik Shellových a JavaScriptových skriptů. Avšak jak tyto skripty jsou využívány pouze pro účely nepřímo spojené se samotným algoritmem (například stažení dat, porovnávání algoritmu s jinými aplikacemi apod.).

7.1 Struktura kódu

Kořenová složka aplikace obsahuje tři nejdůležitější složky – *src, data* a *tests* Složka data byla již popsána v předešlé kapitole, tato kapitola se zaměřuje na složky *src* a *tests*.

Složka *tests* obsahuje experimenty s modely S2V, benchmarky a jednotkové testy. Podsložky začínající textem *unit* obsahují unit testy. Složka *PCA_visualization* obsahuje skripty, které vytvoří model a poté ho vizualizují pomocí PCA. Složky začínající textem *benchmark* obsahují skripty pro

[1] https://www.jetbrains.com/pycharm/

porovnání metody S2V s jinými metody[2]. Tato porovnání jsou jak ohledně rychlosti, tak ohledně přesnosti metod.

Při spouštění těchto skriptů na přesnost predikce, musí ve složce *data* musí být struktury, nad kterými algoritmus poběží. Spouštění skriptů na porovnání rychlosti běží nad daty ze složky *data-test*.

Složka *src* obsahuje zdrojový kód k metodě S2V. Přímo ve složce *src* se nachází skript *main.py*, kterým se spouští samotný kód. Více o spouštění programu lze nalézt v kapitole 7.3.

Ve složce *src* je několik podadresářů, zde je jejich výčet a krátký popis k čemu slouží.

- configuration – obsahuje základní konfiguraci pro vstupní soubory a pro S2V algoritmus
- formatchangers – obsahuje algoritmy pro převod z jednoho formátu na druhá(například *bpseq*bpseq na *db* formát)
- predictors – nejdůležitější část celé aplikace. Tato část obsahuje Struc2Vec algoritmus (trénování modelu, predikce apod.)
- structureloaders – obsahuje algoritmy pro načítání formátu sekundární struktury (například *bpseq* formát)
- utils – obsahuje pomocné funkce pro různé části aplikace
- wordscreators – obsahuje implementaci tvorby slov pro S2V algoritmus

7.2 Vstupní soubory

Tato kapitola popisuje formát vstupních souboru, jak musí být adresáře organizované pro úspěšné natrénování modelu S2V.

Vstupní soubory pro natrénování modelu aplikace je potřeba vložit do složky, která obsahuje složky s příslušnými typy sekundárních struktur RNA. Tedy při spouštění aplikace nad adresářem *data* se dvěma třídami sekundárních

[2]Zejména s aplikací SimTree (http://bioinfo.cs.technion.ac.il/SimTree/).

struktur je potřeba do adresáře data vložit složku s názvem prvního typu a složku s názvem druhého typu. Tyto podsložky poté musí obsahovat validní sekundární struktury RNA v jednom z formátu specifikovaném v kapitole 2.1.2. Přípona jednotlivých struktur musí být dle jeho formátu (tedy *.bpseq*, *.db* nebo *.ct*)

7.3 Spouštění a parametry aplikace

Spouštění aplikace probíhá vždy přes skript *main.py*. Rozhodnutí zdali chce uživatel vytvořit model nebo predikovat data je programu předáváno v podobě parametrů.

Pokud uživatel nezadá žádné parametry, nebo zadá argumenty nesprávné, je do konzole vypsána informace o parametrech.

Seznam možností spuštění aplikace S2V z kořenové složky aplikace, vysvětlivky pod seznamem (argument v závorce znamená, že není povinný, u názvu argumentů stačí vždy použít první písmeno, ale pouze s jednou pomlčkou):

- python3 --train CESTA-K-ADRESÁŘI-S-DATY (--filename NÁZEV-VYTVOŘENÉHO-MODELU)
- python3 --vector CESTA-K-MODELU --data CESTA-K-TESTOVACÍM-DATŮM
- python3 --svm CESTA-K-MODELU --data CESTA-K-TESTOVACÍM-DATŮM
- python3 --plot CESTA-K-MODELU
- python3 --help

Přepínač --train (-t) způsobí, že bude vytvořen model se strukturami obsaženými ve složce za tímto argumentem. Je nutné, aby byla dodržena struktura trénovacích dat dle kapitoly 7.2. Současně s tímto argumentem může být použit argument –filename (-f), který specifikuje název vytvořeného souboru. Pokud není argument specifikující jméno souboru přitomen, je název nově vytvořeného modelu "*S2V.model*".

Přepínač --vector (-v) slouží k predikci dat pomocí vzdáleností mezi jednotlivými vektory v natrénovaném modelu S2V. Za tímto argumentem musí vždy následovat cesta k natrénovanému modelu. Tento způsob predikce vrací nejen typ sekundární struktury, ale také podobnost vektorů. Tento přepínač musí být spuštěn vždy současně s přepínačem --data (-d).

Přepínač --svm (-s) slouží stejně jako přepínač --vector k predikci typu dat. Za tímto argumentem musí vždy následovat cesta k natrénovanému modelu. Tento způsob predikce vrací pouze typ sekundární struktury. Pokud využíváte k predikci svm, je třeba brát v potaz, že svm rozděluje pouze dvě skupiny. Pokud je model natrénovaný na více skupinách, je možno rozeznávat pouze mezi první natrénovanou skupinou a ostatními. Tento typ predikce není doporučen z více důvodů. Prvním důvodem je, že implementace tohoto SVM proběhla z důvodů otestování prostoru vytvořeného pomocí metody S2V a od té doby nebyla moc rozšiřována. Metoda pomocí porovnání vektoru poskytuje lepší výsledky než SVM. V neposlední řadě nelze předpokládat, že vytvořené prostory budou lineárně rozdělitelné (příklad vizualizace prostoru lze nalézt v sekci 8.4).

Přepínač --plot (-p) slouží k vizualizaci natrénovaných dat pomocí PCA metody. Tato metoda lze také použít k manuálnímu hledání dobrých kandidátů, v takovém případě, by bylo potřeba natrénovat algoritmus jak na známých, tak neznámých datech a poté se pomocí vizualizace podívat, které body jsou blízko sebe.

Přepínač --data (-d) specifikuje data, která mají být predikována. Za tímto přepínačem se musí vždy nacházet cesta buďto ke složce se strukturami, které mají být predikovány, nebo k souboru, který má být predikován. Tento přepínač musí být vždy kombinován s přepínačem --vector nebo --svm.

Aplikace přijímá i další argumenty, které specifikují, jakým způsobem se má model natrénovat apod. Tyto argumenty nejsou povinné a uživatel by je měl modifikovat jen v případě, že doopravdy rozumí účelům daných argumentů. Níže je seznam těchto argumentů, za každým argumentem musí následovat číselná hodnota. Více informací o těchto parametrech je popsáno v kapitole 4.2.1.

- --max-epochs – určuje počet epoch trénování modelu S2V. Výchozí hodnota je 100
- --vec-size – určuje počet dimenzí při trénování modelu. Výchozí hodnota je 50

- --alpha – určuje počáteční rychlost učení. Výchozí hodnota je 0.025
- --min-alpha – minimální hodnota rychlosti učení. Výchozí hodnota je 0.00025
- --min-count – určuje minimální počet slov, aby se slovo bralo v potaz. Výchozí hodnota je 1
- --dm – určuje typ algoritmu Doc2vec, který má být použit. Výchozí hodnota je 1
- --learning-decrease – určuje krok snižování hodnoty alpha. Výchozí hodnota je 0.0002
- --distance-bin – určuje velikost "binu"pro vzdálenost při vytváření slov. Výchozí hodnota je 0 (nedochází k redukci počtu slov)
- --length-bin – určuje velikost "binu"pro délku při vytváření slov. Výchozí hodnota je 0 (nedochází k redukci počtu slov)
- --speed – určuje rychlost a přesnost vektor při predikci pomocí vektoru. Může nabývat hodnot 1,2 nebo 3 (ty se mapují do hodnot kroků 50, 250 a 1000). Tento parametr nastavuje počet kroků (z angl. steps) k odhadu vektoru v prostoru. Hodnota 1 je rychlá a nepřesná, zatímco hodnota je pomalá, ale přesná. Výchozí hodnota je 3
- --neighbors – určuje počet sousedů pro predikci nejpodobnější sekundární struktury RNA. Výchozí hodnota je 1.

Kapitola 8

Výsledky

Tato kapitola popisuje výsledky dosažené metodou Struc2Vec. Byly využity výchozí nastavení parametrů metody, pokud není zmíněno, že to tomu bylo jinak. V této kapitole se Struc2Vec porovnává s jinými metodami. Je důležité podotknout, že ostatní metody dokáží porovnat dvě sekundární struktury, aniž by měli jiné struktury na natrénování. Vycházíme ale z toho, že trénink metody Struct2Vec proběhne pouze jednou a čas nutný pro naučení modelu je vzhledem k pozdějšímu využití metody k posouzení podobnosti struktur nad rozsáhlým transkriptomem podstatný. Princip fungování metod je tedy o něco jiný, ale všechny zmíněné metody slouží k porovnávání sekundárních struktur RNA.

Přesnost Struc2Vec byla určována na sekundárních strukturách z databáze CRW a RNase P. Z databáze CRW byly využity sekundární struktury bakterií a eukaryot (jedná se o rozeznávání mezi dvěma typy), zatímco z databáze RNase P bylo využito deset typů sekundárních struktur (jedná se o rozeznávání mezi více typy – v tomto případě mezi desíti).

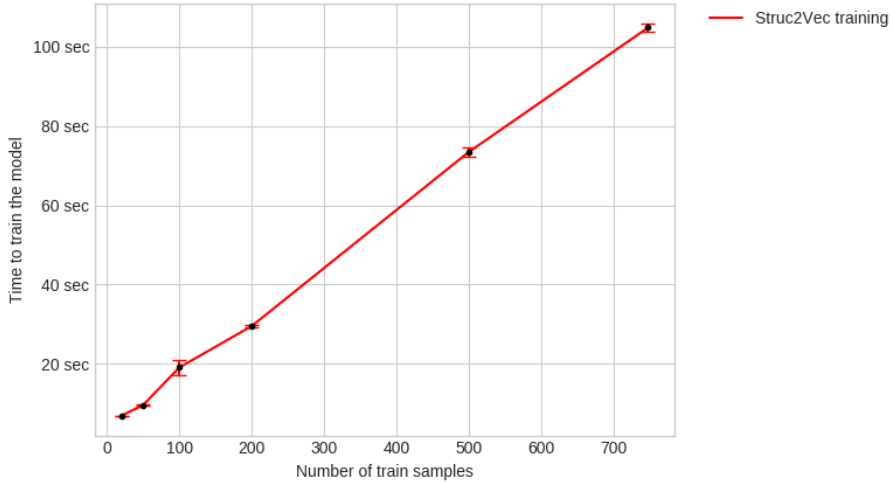
Přesnost predikce je určována pomocí stratifikované křížové validace[1]. Úspěšnost algoritmu je určeno pomocí metody nejbližších k-sousedů v závislosti na reálných výsledcích (typech).

[1] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

8.1 Rychlost Struc2Vec

Tato kapitola popisuje rychlost trénování a predikce algoritmu Struc2Vec. Rychlost trénování je lineárně závislá na počtu sekundárních struktur v trénovacích datech. Delší struktury také znamenají pomalejší trénování. Pro trénování (predikci) byly využity data z CRW databáze. Délka jednotlivých sekvencí je různá, ale neliší se výrazně mezi jednotlivými experimenty. Rychlost metody S2V je také závislá na počtu kroků pro odhad vektoru testovaných dat, tato závislost je popsaná v sekci 8.2.

Z grafu 8.1 lze vysledovat, že doba trénování modelu je lineární vzhledem k počtu struktur v trénovacích datech.



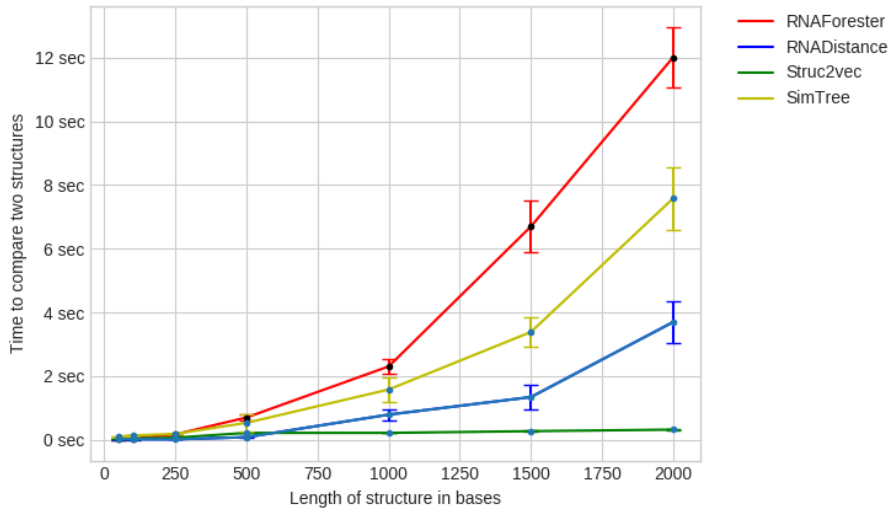
Obrázek 8.1: Doba trvání trénování modelu v závislosti na počtu sekundárních struktur

Dále jsme se zabývali samotnou dobou srovnání podobnosti dvou sekundárních struktur v závislosti na jejich délce. Na obrázku 8.2 lze vidět porovnání rychlosti algoritmu Struc2Vec, SimTree, RNAForesteru (balíček RNAVienna) a RNADistance (balíček RNAVienna). Struc2Vec je výrazně rychlejší než ostatní aplikace, a navíc složitost metody roste s délkou porovnávaných sekvencí v lineárním čase, oproti ostatním aplikacím, kde empirická složitost odpopvídá teoretickým předpokladům z kapitoly 3.2.

Aplikace RNADistance je rychlejší než aplikace SimTree nejspíš z důvodů, že implementace je v jazyce C++, zatímco implementace aplikace SimTree je v jazyce Java. Program RNAForester je pomalejší než ostatní testované aplikace, ale dokáže porovnat více struktur najednou. Při porovnávání více struktur najednou je ale čas výpočtu výrazně vyšší. Na mém počítači se

mi nepodařilo spustit program RNAForester na větším množství struktur, nejspíš z důvodu nedostatku RAM.

Rychlost metody Struc2Vec byla počítána zanořením dvou sekundárních struktur do vektorového prostoru vytvořeného modelu. Důvodem pro zanořování dvou sekundárních struktur namísto jedné je férové porovnání s ostatními metodami. Ostatní metody mají na vstupu vždy dvě sekundární struktury, které porovnávají najednou. Rychlost při zanoření pouze jedné struktury do vektorového prostoru modelu by byla přibližně poloviční.



Obrázek 8.2: Porovnání časové složitosti algoritmů pro porovnání sekundárních struktur

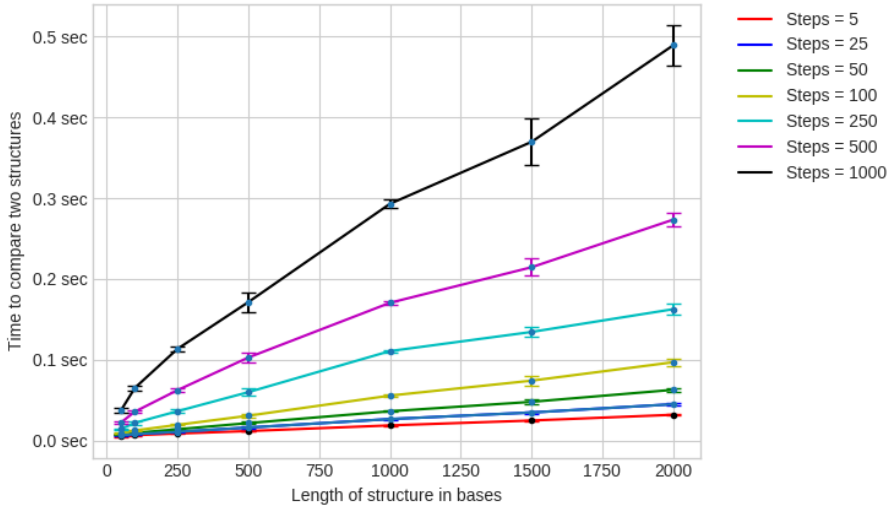
8.2 Porovnání rychlosti ku přesnosti

Tato sekce popisuje, jak moc je algoritmus ovlivněn parametrem pro počet kroků.

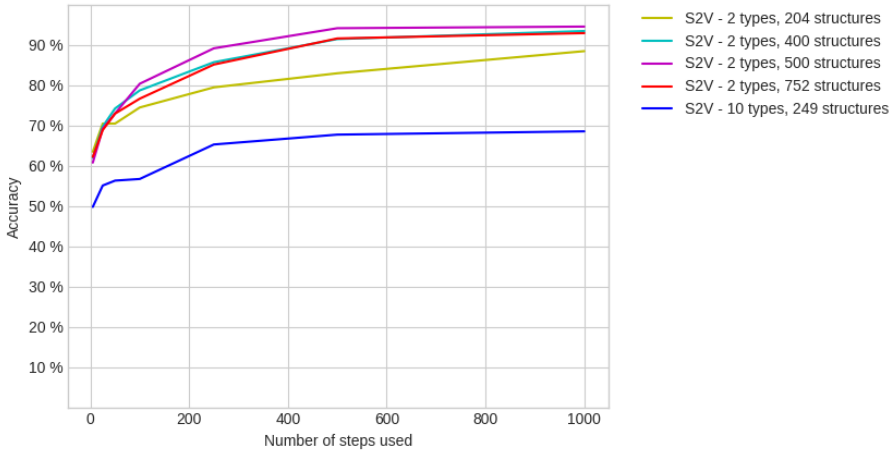
Doba predikce vektoru sekundární struktury je lineárně závislá na délce struktury. Tato rychlost je také ovlivněna parametrem "počet kroků". Čím vyšší počet kroků, tím je algoritmus pomalejší, ale přesnější (více informací v sekci 4.2.1). Na grafech 8.3 a 8.4 lze vidět, jak moc tento parametr ovlivňuje výsledky modelu.

Z grafů lze vidět, že rychlost algoritmů stoupá lineárně s počtem kroků. Přesnost výpočtu stoupá ze začátku výrazně a poté konverguje k určité

hodnotě. V závislosti na těchto výsledcích byl zvolen počet kroků 1000 pro predikci typu sekundární struktury.



Obrázek 8.3: Vliv počtu kroků na dobu predikce struktury



Obrázek 8.4: Vliv počtu kroků na přesnost predikce struktury

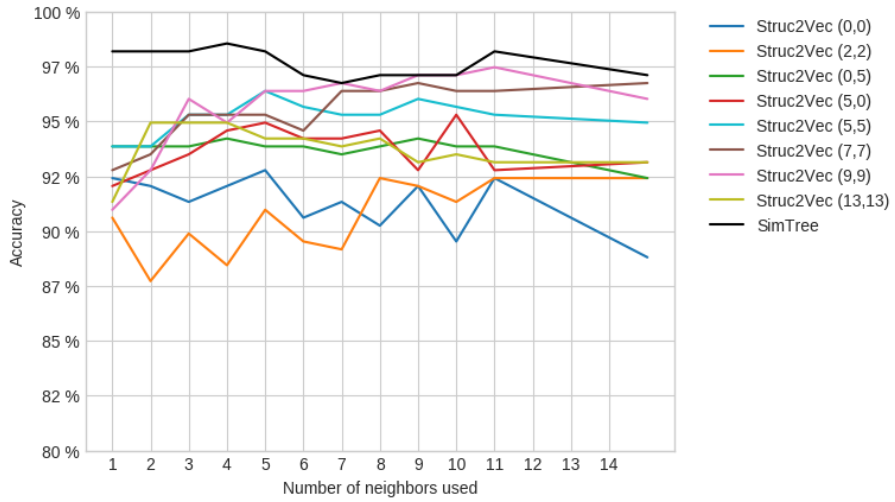
8.3 Vliv redukce počtu slov na přesnost predikce

Tato sekce popisuje výsledky dosažené pomocí metody redukce počtu slov, která byla popsána v sekci 5.1.1. Nejprve jsou vidět výsledky při porovnávání dvou typů struktur a poté následují výsledky při porovnávání deseti typů struktur.

Na obrázcích 8.5 a 8.6 lze vidět vliv redukce slov na úspěšnost metody

Struc2Vec. Z obrázků lze vidět, že čím více struktur je k dispozici, tím menší význam ma redukovat počet slov. Při využití 277 struktur bylo ideální použít redukci slov podle vzdálenosti i délek na 7. Při učení na 748 je ideální redukovat slova podle vzdálenosti i délek na 2. Samotná redukce podle jednoho z těchto dvou parametrů taky pomáhala, ale výsledkem testů bylo, že omezení obou na stejnou délku vede zpravidla k lepším výsledkům.

Je důležité podotknout, že aplikace SimTree byla spouštěna jen na kratších sekundárních strukturách, protože i přes doporučení na stránkách SimTree[2], došlo občas při porovnávání struktur k chybě a výsledky tím pádem byly ovlivněny (docházelo k horším výsledkům, než by reálně mělo). Pro aplikaci SimTree bylo zvoleno 277 sekundárních struktur, které byly otestovány i na metodě S2V. Těchto 277 struktur bylo vybráno podle délky, tedy struktury delší než 1400 bází byly z data setu odstraněny. Porovnání při trénování na 748 strukturách je tedy pouze přibližné, reálné výsledky SimTree pro 748 struktur nejsou dostupné.

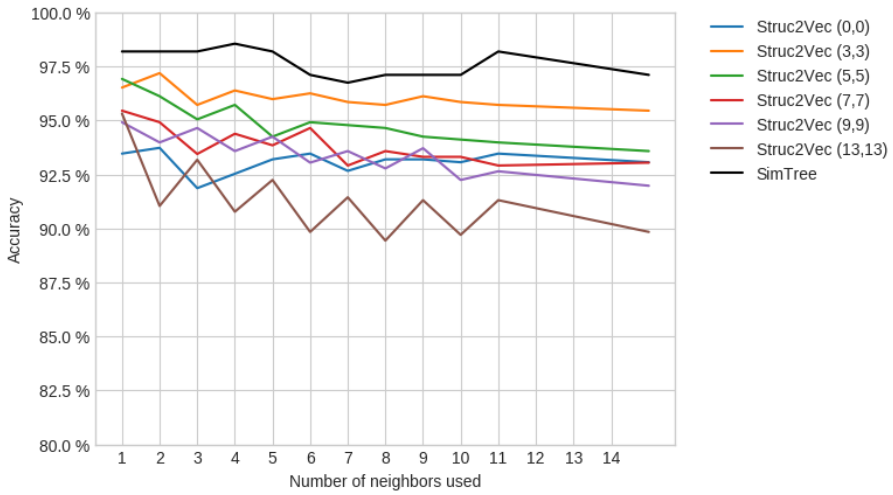


Obrázek 8.5: Výsledky při porovnání 277 struktur dvou typů

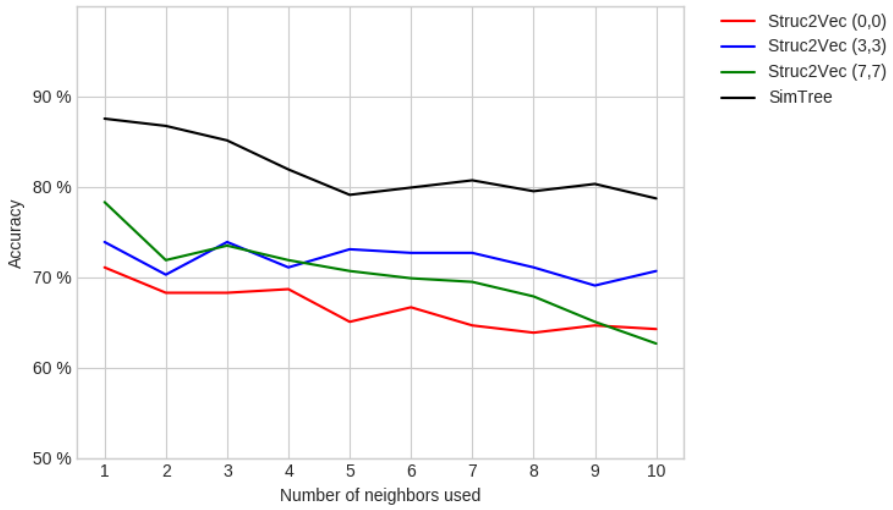
Na grafu 8.7 jde vidět vliv redukce počtu slov pro přesnost predikce na 10 typech struktur. V nejlepším případě (redukce vzdáleností a délek na 7), bylo dosaženo přesnosti 78.3 %. U aplikace Simtree bylo dosaženo přesnosti 87.6 %. Tedy rozdíl v přesnosti je 9 %. Při porovnání více sousedů byl rozdíl v přesnosti ještě o něco nižší. Opět lze vidět, že redukce počtu slov pomohla k lepším výsledkům. Důvodem je opět nejspíš příliš malý trénovací korpus.

Z grafů lze vyčíst, že metoda redukce počtu slov funguje dobře při příliš malém korpusu trénovacích dat. Problémem je volba parametrů redukce počtu slov, jelikož ideální volba parametru by musela brát v potaz počet struktur,

[2]http://bioinfo.cs.technion.ac.il/SimTree/



Obrázek 8.6: Výsledky při porovnání 748 struktur dvou typů



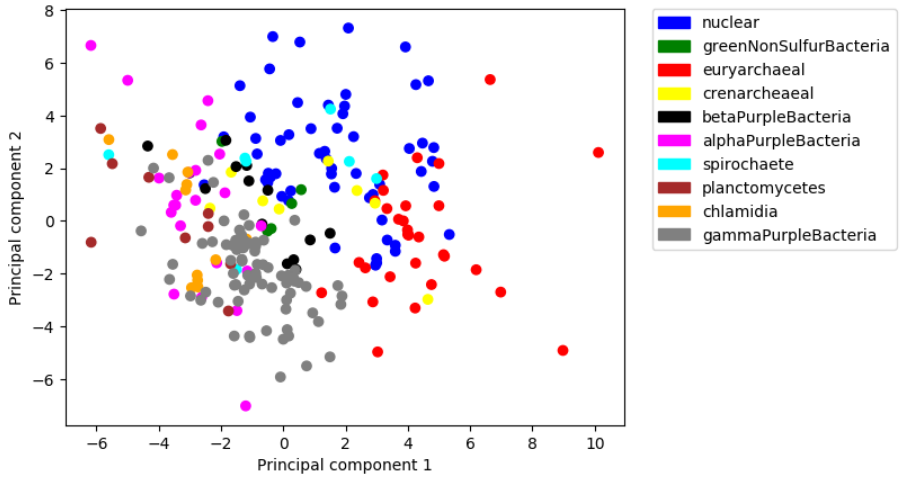
Obrázek 8.7: Výsledky při porovnání 249 struktur deseti typů

délku struktur, počet částí struktur, typy částí struktur a další vlastnosti trénovacího korpusu. Také lze vyčíst, že přesnost výpočtu se přibližuje přesnosti výpočtu pomocí tradičních metod, avšak pro velmi dobrou přesnost algoritmu Struc2Vec je potřeba vhodně zvolit parametry redukce počtu slov.

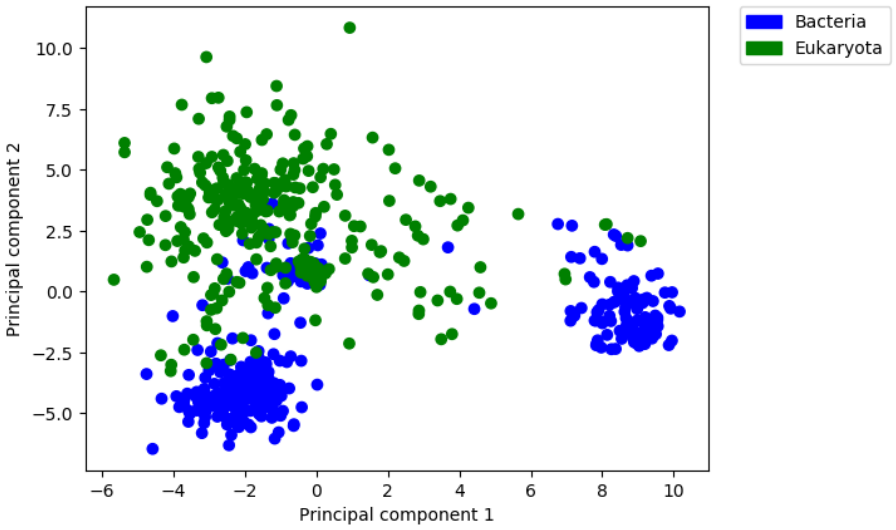
8.4 Vizualizace prostoru Struc2Vec

Tato sekce obsahuje vizualizace prostorů vytvořených metodou Struc2Vec. Pro vizualizaci byla použita metoda PCA. Vizualizované vzdálenosti mezi body nemusí stoprocentně odpovídat skutečné vzdálenosti ve vytvořeném prostoru, protože v procesu snižování dimenzí je určitá část informací ztracena.

Na obrázku 8.8 lze vidět, že při vizualizaci více typů RNA struktur se mnoho typů struktur překrývá, avšak určité shluky jednotlivých typů lze nalézt. Na obrázku 8.9 lze vidět, že sekundární struktury dvou typů (eukaryot a bakterií) se velmi dobře oddělily od sebe.



Obrázek 8.8: Vizualizace prostoru pro 10 různých typů struktur (získaných z databáze RNase P).



Obrázek 8.9: Vizualizace prostoru pro 2 typy struktur (získaných z databáze CRW).

8.5 Využití na biologických datech

Aplikace byla předána kolegovi Anh Vu Le Quy z univerzity ČVUT, z elektrotechnické fakulty, z důvodů využití na jeho diplomové práci, která vznikala ve stejném období jako tato práce. Aplikace by měla být využita při porovnávání sekundárních RNA struktur IRES Hepatitidy C. Dále byla také aplikace natrénována na strukturách zmíněného IRES a poté byly predikovány neznámé struktury. Výsledky u predikce neznámých struktur nebyly přesvědčivé. Byly natrénovány dva modely, jeden obsahující pouze data IRES, druhý obsahují kromě dat IRES také struktury z databáze RNase P. Pro každý model byly vytvořeny dva soubory, jeden obsahoval podobnost seřazenou podle průměrné vzdálenosti od všech IRES a druhá obsahoval podobnost seřazenou podle největší podobnosti.

Pro model obsahující pouze IRES vyšel docela dobrý výsledek (18. pozice z 1000) v případě porovnání podle největší podobnosti (porovnání dle 1 nejbližšího souseda). Pro model obsahující struktury IRES i struktury z databáze RNase P vyšel dobrý výsledek (20. pozice z 1000) pro porovnání průměrné vzdálenosti od všech IRES. Ostatní výsledky (model natrénován pouze z IRES – průměrná vzdálenost a model natrénován z IRES a dat z RNase P – největší podobnost) byly dle kolegy Anh Vu Le Quy špatné, hit se nenacházel v pořadí na horních příčkách.

K účelům detekce IRES hepatitidy C v člověku lze možná použít model

trained_all_biological.s2v, který se nachází ve složce *models*. Tento model byl natrénován na velkém množství struktur a v případě, že by se struktura porovnávaná s tímto modelem velmi podobala strukturám IRES hepatitidy C, dala by se daná struktura považovat za kandidátní výskyt. Také byl vytvořen model *ires.s2v*, který natrénován na sekundárních strukturách IRES hepatitidy C. Tyto sekundární struktury byly dodány kolegou Anh Vu Le Quy.

8.6 Shrnutí výsledků a diskuze

V této kapitole jsou shrnuty výsledky dosažené nově vytvořeným algoritmem Struc2Vec. Tyto výsledky jsou porovnávány s metodou výpočtu editační vzdálenosti stromů.

Tabulka 8.1 popisuje časy porovnání dvou struktur v případě délek bází 500 a 2000. Metoda Struc2Vec dosahuje výrazného zlepšení rychlosti v případě, že jsou porovnávány dlouhé struktury RNA. V případě krátkých struktur RNA je rozdíl méně znatelný a v určitých případech je při porovnání dvou sekvenci metoda Struc2Vec pomalejší. V případě, že je cílem porovnat více struktur najednou, je metoda Struc2Vec rychlejší i při porovnávání krátkých struktur a to z důvodu, že Struc2Vec dokáže porovnat jednu strukturu s velkým množství natrénovaných struktur najednou.

Název aplikace	Rychlost (50 bází)	Rychlost (500 bází)	Rychlost (2000 bází)
SimTree	$0,10 \pm 0,01$	$0,54 \pm 0,27s$	$7,58 \pm 0,99s$
RNADistance	$0,03 \pm 0,02$	$0,08 \pm 0,02s$	$3,70 \pm 0,65s$
RNAForester	$0,02 \pm 0,02$	$0,70 \pm 0,10s$	$12,0 \pm 0,95s$
Struc2Vec	$0,03 \pm 0,001$	$0,22 \pm 0,002s$	$0,32 \pm 0,01s$

Tabulka 8.1: Tabulka rychlosti porovnání pomocí různých metod

Metoda Struc2Vec dosahuje lineární časové složitosti, což odpovídá výsledkům zobrazených v tabulce 8.1 a výsledkům z kapitoly 8. Porovnání časových složitostí současných metod a metody Struc2Vec lze nalézt v tabulce 8.2.

Výpočet editační vzdálenosti stromů	$O_{time}(N^3)$ [13]
Levenshteinova vzdálenost	$O_{time}(N^2)$ [30]
Výpočet založený na metodě Lempel-Ziv	$O_{time}(N^2)$ [31]
Struc2Vec	$O_{time}(N)$

Tabulka 8.2: Tabulka složitostí algoritmů porovnávajících sekundární struktury RNA

Pro porovnání korelace metod Struc2Vec a editační vzdálenosti stromů

(využita byla aplikace SimTree) byl proveden výpočet Spearmanova koeficientu pořadové korelace[3], která bere v potaz pořadí výsledků. Dalé byl proveden výpočet Pearsonova korelačního koeficientu[4], který bere v potaz korelaci samotných hodnot.

Při výpočtu korelace bylo porovnáváno 277 sekundárních struktur dvou typů (*bacteria* a *eukaryota*) z CRW databáze. Výsledky korelace lze vidět v tabulce 8.3. U Struc2Vec bylo využito binování podle délek i vzdálenosti o velikosti 9.

Výsledky korelačních koeficientů jsou zpravidla relativně nízké a mají velkou standardní odchylku. Tento jev může být způsoben tím, že mnoho hodnot je velice podobných (jak při porovnávání pomocí SimTree, tak při porovnávání pomocí S2V). Například u jednoho vzorku se hodnota 0.30 vyskytuje hned dvacetkrát (z 277 hodnot, u výsledků editační vzdálenosti). Tyto podobné hodnoty mohou silně zkreslovat výsledky.

Název metody	Průměr	Medián	Maximum	Minimum
Spearman	$0,41 \pm 0,43$	$0,42$	$0,86$	-0.47
Pearson	$0,49 \pm 0,39$	$0,53$	$0,92$	-0.27

Tabulka 8.3: Tabulka korelace metod S2V a editační vzdálenosti na data setu CRW při použití 277 struktur

Výsledky přesnosti metody S2V byly validovány na dvou data setech získaných z databází CRW a RNase P. Byla otestována přesnost metody na dvou typech struktur (CRW) a na deseti typech struktur (RNase P). Přesnost metody nedosahuje kvality výsledků klasické metody porovnání editační vzdálenosti. V tabulce 8.4 lze vidět souhrn obou metod na zmíněných data setech. Procenta zmíněná v tabulce jsou dosažena pomocí tří nejbližších sousedů. Více informací o přesnosti metody je zmíněno v sekci 8.3.

Název metody	CRW data, 277 struktur	CRW data, 748 struktur	RNase P data
SimTree	$98,2\%$	——	85.1%
Struc2Vec (0,0)	$91,3\%$	$91,9\%$	$68,3\%$
Struc2Vec (3,3)	$95,7\%$	$95,7\%$	$73,9\%$
Struc2Vec (7,7)	$95,3\%$	$93,4\%$	$73,5\%$

Tabulka 8.4: Tabulka přesnosti porovnání sekundárních struktur

Horší výsledky než u tradičních metod byly očekávány. Na obrázcích v příloze B lze vidět několik vizualizovaných příkladů, které byly vizualizovány pomocí aplikace VARNA [12]. U některých struktur, které S2V klasifikoval podobně lze vidět podobnost sekundárních struktur od oka. Avšak v některých případech podává S2V výsledky, které vypadají od oka špatně (takový

[3] https://cs.wikipedia.org/wiki/Spearman%C5%AFv_koeficient_po%C5%99adov%C3%A9_korelace
[4] https://www.wikiskripta.eu/w/F%C3%B3rum:Testy/Pearson%C5%AFv_korela%C4%8Dn%C3%AD_koeficien

příklad jde vidět na obrázku B.3b). Důvodem proč hodnotí S2V od oka rozdílné struktury jako podobné, může být kombinace podobnosti malé části struktury a malého množství slov při učení algoritmu. Zmíněný příklad není zcela ojedinělý, zatímco u metody porovnání pomocí editační vzdálenosti se tento jev nevyskytuje (nepovedlo se jej najít v testovacích datech).

Důvodem velmi rozdílné klasifikace oproti porovnání pomocí editační vzdálenosti může být to, že S2V nebere v potaz délku sekvence, hledá jen společné části. Při počítání editační vzdálenosti je délka sekvence přece jen brána v potaz – v případě, že jedna struktura je delší než druhá, je editační vzdálenost mezi těmito strukturami alespoň rozdílem jejich délek (více informací v sekci 3.2).

Redukce počtu slov (binování) je užitečná v případě nedostatečně velkého trénovacího korpusu. Více informací o redukci počtu slov lze nalézt v sekci 8.3.

Kapitola 9

Závěr

V této práci byla implementována nová metoda Struc2Vec pro porovnávání sekundárních struktur RNA. Metoda byla otestována na sekundárních RNA strukturách převzatých z online databází. Největší výhodou nově vytvořené metody je především její rychlost, která je lineární v závislosti na délce sekundární struktury. Konkurenční metody využívající klasické přístupy k výpočtu podobnosti struktury pracují v kubickém čase. Tento rozdíl v rychlosti je obrovský, hlavně z důvodu, že při práci s RNA je často potřeba porovnávat velké množství dlouhých řetězců (například při práci s lidskou RNA).

Mezi hlavní nevýhody Struc2Vec patří především nutnost mít trénovací data, na kterých se algoritmus bude naučit, jak vypadají struktury určitého typu. Ostatní metody, se kterými byl algoritmus porovnáván, nemají žádnou trénovací fázi, ale pouze porovnávají dvě vstupní struktury. Struc2Vec taktéž nedokáže porovnávat dvě vstupní sekvence, nýbrž každá vstupní sekvence je porovnána pouze s natrénovanými daty.

Další nevýhodou Struc2Vec je menší přesnost u porovnávání struktur. Při klasifikování mezi deseti typy struktur dosáhl algoritmus Struc2Vec přesnosti přibližně 74 %, zatímco konkurenční algoritmy dosáhly přesnosti přibližně 85 %. U klasifikování mezi dvěma typy struktur bylo Struc2Vec algoritmem dosaženo skoro 96% úspěšnosti, zatímco konkurenčními algoritmy dosáhly více než 98% úspěšnosti.

Vytvořený algoritmus je možné používat i na jiné sekundární struktury, než jsou struktury RNA. Tato práce se zabývala zejména sekundárními strukturami RNA, ale neexistuje důvod, proč by se nedala práce rozšířit

i na DNA, proteiny, či jiné látky s dostupnou sekundární strukturou ve stejných tvarech jako má RNA.

Práce poskytuje určitý prostor pro pokračování, například je možné vytvářet mnoho dimenzionální prostor pomocí jiné metody než Doc2vec. Takovou metodou by mohly být například hluboké neuronové sítě. Také predikce typu RNA lze provádět jinak, než na základě vzdáleností, je možné využít například více třídní SVM. Prostor pro rozšíření by mohl být také při nastavování parametrů, které by šlo nastavovat dynamicky v závislosti na trénovacích datech.

V budoucnu by se také dal vytvořit velký model obrovského množství sekundárních struktur (i jiných typů než RNA) a vytvořit volně dostupnou webovou stránku, kde by si mohli uživatelé porovnávat své RNA struktury s velkým množstvím natrénovaných RNA struktur. Aby tato práce byla dostupnější pro větší spektrum uživatelů, je dle mého úsudku toto velmi dobrým prvním krokem.

Přílohy

Příloha A

Literatura

[1] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[2] S. Abhishek. What is the differences between artificial neural network (computer science) and biological neural network? Převzato z https://www.quora.com/What-is-the-differences-between-artificial-neural-network-computer-science-and-biological-neural-network.

[3] M. A. Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[4] S. Altschul and B. Erickson. Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology*, 48(5-6):603–616, 1986.

[5] H. Berghel and D. Roach. An extension of ukkonens enhanced dynamic programming asm algorithm. *ACM Transactions on Information Systems*, 14(1):94–106, 1996.

[6] J. W. Brown. The Ribonuclease P Database. *Nucleic Acids Research*, 26(1):351–352, 01 1998.

[7] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Müller, and et al. The comparative rna web (crw) site: an online database of comparative sequence and structure information for ribosomal, intron, and other rnas.

[8] M. Caudill. Neural networks primer, part i. *AI Expert*, 2(12):46–52, Dec. 1987.

[9] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants, Apr 2010.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Sept. 1995.

[11] V. Cvrček. Portál. Dostupné na https://www.korpus.cz/.

[12] K. Darty, A. Denise, and Y. Ponty. Varna: Interactive drawing and editing of the rna secondary structure. *Bioinformatics*, 25(15):1974–1975, 2009.

[13] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1):1–19, 2009.

[14] Y. Ding and C. Lawrence. A statistical sampling algorithm for rna secondary structure prediction. *Nucleic acids research*, 31:7280–301, 01 2004.

[15] E. Domingo and J. J. Holland. Rna virus mutations and fitness for survival. *Annual Review of Microbiology*, 51(1):151–178, 1997. PMID: 9343347.

[16] E. Eden, I. Wallach, and Z. Yakhini. dostupné na http://bioinfo.cs.technion.ac.il/SimTree/Supplementary_material_for_the_web.htm.

[17] R. Eliáš and D. Hoksza. RNA secondary structure visualization using tree edit distance. *International Journal of Bioscience, Biochemistry and Bioinformatics*, 6(1):9–17, 2016.

[18] C. Gaurav. What is the difference between deep and shallow neural networks?

[19] R. M. Golden. *Mathematical Methods for Neural Network Analysis and Design*. MIT Press, Cambridge, MA, USA, 1st edition, 1996.

[20] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162(3):705–708, 1982.

[21] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[22] Z. S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954.

[23] D. L. Hartl and M. Ruvolo. *Genetics: analysis of genes and genomes*. Jones & Bartlett Learning, 2012.

[24] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[25] I. Hofacker, D. Steininger, and S. Findeiß. Tbi - theoretical biochemistry group, Aug 2017. dostupné na `https://www.tbi.univie.ac.at/RNA/tutorial/`, navštíveno 2.3.2019.

[26] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte fuer Chemie Chemical Monthly*, 125(2):167–188, 1994.

[27] M. Höchsmann, B. Voss, and R. Giegerich. Pure multiple rna secondary structure alignments: A progressive profile approach. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 1:53–62, 01 2004.

[28] P. N. Klein. Computing the edit-distance between unrooted ordered trees, 1998.

[29] Q. Le and T. Mikolov. Distributed representations of sentences and documents, May 2014.

[30] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, Feb. 1966.

[31] N. Liu and T. Wang. A method for rapid similarity analysis of rna secondary structures. *BMC Bioinformatics*, 7(1):493, Nov 2006.

[32] Z. S. e. a. N. Y. W. H. F. Lodish H, Berk A. *Molecular cell biology*. New York: W.H. Freeman and CO., 2000. ISBN: 978-0-7167-4366-8.

[33] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*, 6(1):26, Nov 2011.

[34] F. Madzharova, Z. Heiner, M. Gühlke, and J. Kneipp. Surface-enhanced hyper raman spectra of adenine, guanine, cytosine, thymine, and uracil. *The Journal of Physical Chemistry C*, 120, 06 2016.

[35] A. D. McNaught and A. Wilkinson. *IUPAC. Compendium of Chemical Terminology*. the "Gold Book". Blackwell Scientific Publications, 2 edition, 1997.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, Sep 2013.

[37] P. B. Moore. Structural motifs in rna. *Annual Review of Biochemistry*, 68(1):287–300, 1999. PMID: 10872451.

[38] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. In S. Brenner, editor, *Molecular Biology*, pages 453 – 463. Academic Press, London, 1989.

[39] A. neznámý. Cis520 machine learning | lectures / pca. převzato z https://alliance.seas.upenn.edu/ cis520/wiki/index.php?n=Lectures.PCA.

[40] A. neznámý. Nature news - mutation. Nature Publishing Group, převzato z https://www.nature.com/scitable/definition/mutation-8.

[41] A. neznámý. Orders of protein structure. Khan Academy, dostupné na https://www.khanacademy.org/science/biology/macromolecules/proteins-and-amino-acids/a/orders-of-protein-structure.

[42] A. neznámý. Ribonukleové kyseliny - rna. dostupné na http://web2.mendelu.cz/af_291_projekty2/vseo/print.php?page=1708&typ=html.

[43] A. neznámý. Rna structure formats. dostupné na http://projects.binf.ku.dk/pgardner/bralibase/RNAformats.html.

[44] A. neznámý. Rna, Jan 2018. Wikimedia Foundation, dostupné na https://cs.wikipedia.org/wiki/RNA.

[45] A. neznámý. Pseudoknot, May 2019. Wikimedia Foundation, dostupné na https://en.wikipedia.org/wiki/Pseudoknot.

[46] F. Pappalardo, C. Calonaci, M. Pennisi, E. Mastriani, and S. Motta. Hamfast: Fast hamming distance computation. volume 1, pages 569–572, 01 2009.

[47] M. Pawlik and N. Augsten. Efficient computation of the tree edit distance. *ACM Transactions on Database Systems (TODS)*, 40(3), 1996.

[48] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1511.06434, Nov 2015.

[49] S. Raschka. Kernel tricks and nonlinear dimensionality reduction via rbf kernel pca, Sep 2014. dostupné na https://sebastianraschka.com/Articles/2014_kernel_pca.html.

[50] X. Rong. word2vec Parameter Learning Explained. *arXiv e-prints*, page arXiv:1411.2738, Nov 2014.

[51] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[52] W. S. Sarle. Neural networks and statistical models, 1994.

[53] B. A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Bioinformatics*, 6(4):309–318, 10 1990.

[54] S. Smit, K. Rother, J. Heringa, and R. Knight. Accepted data formats, 2007. dostupné na http://www.ibi.vu.nl/programs/k2nwww/static/data_formats.html.

[55] S. Smit, K. Rother, J. Heringa, and R. Knight. From knotted to nested rna structures: A variety of computational methods for pseudoknot removal. *RNA*, 14(3):410–416, 2008.

[56] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[57] W.-K. Sung. *Algorithms in bioinformatics: a practical introduction.* Chapman & Hall/CRC, 2010.

[58] K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433, 1979.

[59] J. Venter, M. Adams, E. W Myers, P. Li, R. J Mural, G. Sutton, H. O Smith, M. Yandell, C. A Evans, R. A Holt, J. D Gocayne, P. Amanatides, R. Ballew, D. Huson, J. Wortman, Q. Zhang, C. Kodira, X. Zheng, L. Chen, and X. Zhu. The sequence of the human genome. *Science*, 291:1304–1351, 01 2001.

[60] J. H. Wilson and T. Hunt. *Molecular biology of the cell, 4th edition: a problems approach.* Garland Science, 2002.

[61] H. Xu. An algorithm for comparing similarity betweentwo trees: Edit distance with gaps, Apr 2014.

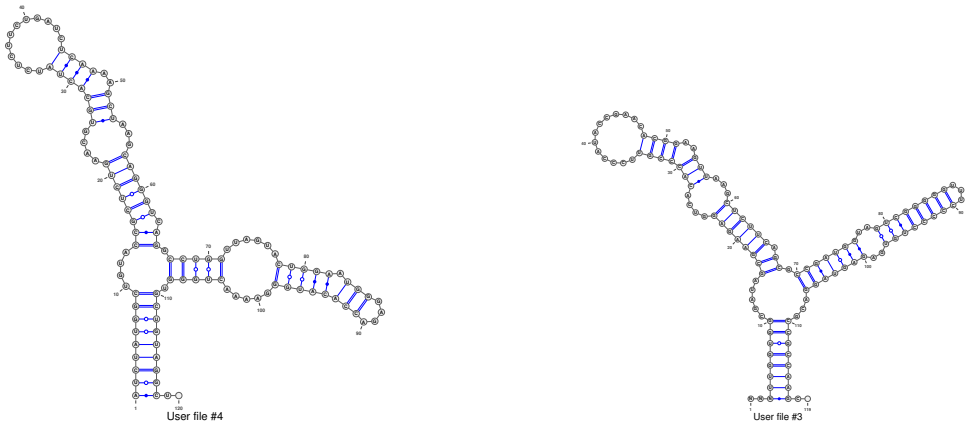
[62] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18:1245–1262, 12 1989.

[63] R. Řehůřek. gensim: topic modelling for humans.

Příloha B

Příklady porovnaných struktur

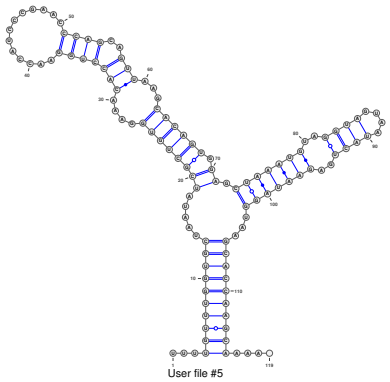
Tato příloha obsahuje příklady struktur, které byly metodou Struc2Vec odhadnuty jako podobné, případně rozdílné. Samotná podobnost je napsaná u jednotlivých obrázků. Popisy obrázku obsahují název organismu a přístupové číslo struktury.



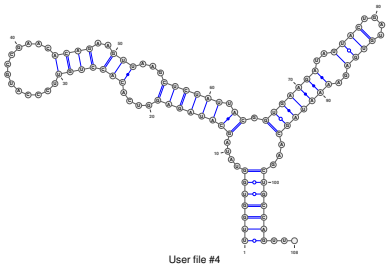
(a) : Homo sapiens – Z75742 **(b) :** Bacillus subtilis – D11460

Obrázek B.1: Příklad struktur predikovaných jako podobné, ale patřících do jiné skupiny [0].

[0] Sekundární struktury byly vizualizovány pomocí aplikace http://varna.lri.fr/

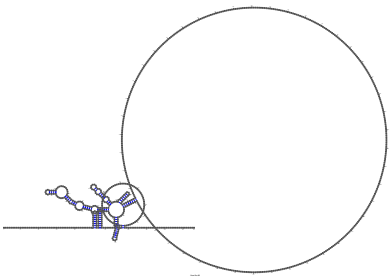


(a) : Mycoplasma genitalium – U39694

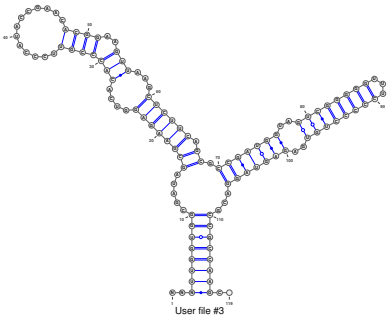


(b) : Mycoplasma capricolum – V00646

Obrázek B.2: Příklad struktur predikovaných jako podobné a patřících do stejné skupiny [0].



(a) : Acanthamoeba castellanii – U03732



(b) : Staphylococcus aureus – L36472

Obrázek B.3: Příklad struktur predikovaných jako podobné, ale vypadajících rozdílně. [0].

Příloha C

Obsah přiloženého CD

K této práci je přiloženo CD, které obsahuje následující složky:

- složka *CODE* obsahuje kód aplikace, který byl popsán v sekci 7.1
- složka *DOCUMENTATION* obsahuje pdf soubor diplomové práce a také zdrojové soubory LaTeX