РЕФЕРАТ

Роботу виконано на 70 аркушах, вона містить 1 додаток та перелік посилань на використані джерела з 22 найменувань.

Метою дипломної роботи є побудова нової моделі природних мов, що дасть змогу якісно використовувати структуру мов для методів криптоаналізу.

Об'єктом дослідження є формальна модель природних мов.

Предметом дослідження є побудова нової моделі природної мови на основі стохастичної контекстно-вільні граматики в нормальній формі Грейбах.

В роботі зроблено огляд класичних і сучасних підходів до моделювання мови, проведено паралелі з іншими галузями науки, де використовується теорія мов та граматик. Запропоновано нову модель відкритого тексту(природної мови) на основі прихованої моделі Маркова другого порядку за спостереженнями зі стеком. Досліджено її властивості і доведена еквівалентність розробленої моделі та стохастичних контекстно-вільних граматик. Запропоновано поліноміальний алгоритм зведення стохастичної контекстно-вільної граматики до введеної моделі. Розроблено програмний пакет з реалізацією запропонованої моделі та емпіричним шляхом підтверджено узгодженість побудованої моделі та стохастичних контекстно-вільних граматик.

Запропонована модель вимагає розробки нових алгоритмів навчання стохастичних граматик.

СТОХАСТИЧНА КОНТЕКСТНО-ВІЛЬНА ГРАМАТИКА, ПРИХОВАНА МОДЕЛЬ МАРКОВА, МОДЕЛЬ ВІДКРИТОГО ТЕКСТУ, ПРИРОДНІ МОВИ

РЕФЕРАТ

Дипломная работа выполнена на 70 листах, она содержит 1 приложение и список ссылок на использованные источники с 22 наименований.

Целью дипломной работы является построение новой модели естественных языков, что дает возможность качественно использовать структуру языков для методов криптоанализа.

Объектом исследования является формальная модель естественных языков.

Предметом исследования является построение новой модели естественного языка на основе стохастических контекстно-свободных грамматик в нормальной форме Грейбах.

В работе проведен обзор классических и современных подходов к моделированию языка, проведены параллели с другими областями науки, где используется теория языков и грамматик. Предложено новую модель открытого текста(естественного языка) основанную на скрытой модели Маркова второго порядка по наблюдениям со стеком. Исследовано ее свойства и доказана эквивалентность разработанной модели и стохастических контекстно-свободных грамматик. Предложено полиномиальный алгоритм сведения стохастической контекстно-свободной грамматики к введенной модели. Разработан программный пакет с реализацией предложенной модели и эмпирическим путем подтверждена согласованность построенной модели с стохастическими контекстно-свободными грамматиками.

Предложенная модель требует разработки новых алгоритмов обучения стохастических грамматик.

СТОХАСТИЧЕСКАЯ КОНТЕКСТНО-СВОБОДНАЯ ГРАММАТИКА, СКРЫТАЯ МОДЕЛЬ МАРКОВА, МОДЕЛЬ ОТКРЫТОГО ТЕКСТА, ЕСТЕСТВЕННЫЕ ЯЗЫКИ

ABSTRACT

The thesis is presented in 70 pages. It contains 1 appendix and bibliography of 22 references.

The target of the thesis is construction of an effective model of natural languages which allows using a structure of language for cryptanalysis more efficiently.

The object is a formal model of natural languages.

The subject is developing a new model of a natural language based on stochastic context-free grammars in Greibach normal form.

We made an overview of classical and modern approaches for language modeling and drew parallels to other spheres of science where a theory of languages and grammars is used. We developed the probabilistic model of stochastic context-free language based on hidden Markov model of the second order with stack and proved equivalence between them. Also, we proposed the polynomial algorithm for transformation of a stochastic context-free grammar to the proposed model. We developed the software which implements proposed model and empirically confirmed concordance between proposed model and stochastic context-free grammars.

Proposed model requires developing of new learning algorithms.

STOCHASTIC CONTEXT-FREE GRAMMAR, HIDDEN MARKOV MODEL, PLAINTEXT MODEL, NATURAL LANGUAGES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\otimes$ — тензорний добуток

π — початковий розподіл ланцюга Маркова

$P(A)$ — імовірність події A

$X_{1:t}$ — вектор $(X_1, X_2, \ldots, X_t)$

$P(X_t = i|X_{1:t})$ — умовний розподіл випадкового елемента X_t при умові сигма-алгебри породженої вектором $(X_1, X_2, \ldots, X_t)$.

ВСТУП

Актуальність дослідження. Бурхливий розвиток штучного інтелекту сьогодні стимулює дослідження структури таких об'єктів, як природні мови, ланцюжки ДНК та інші структури, що мають певну організованість на вищому рівні. Дослідження властивостей та побудова ефективних моделей мови є важливим аспектом криптоаналізу, оскільки це може дати криптоаналітику додаткову інформацію, щодо структури відкритого тексту, до якого застосовуються певні криптографічні перетворення.

Метою дослідження є побудова нової моделі природних мов, що дасть змогу краще використовувати структуру мов для методів криптоаналізу. Для досягнення мети необхідно вирішити наступні **завдання дослідження**:

1) провести огляд опублікованих робіт за тематикою дослідження;

2) розробити модель стохастичних регулярних граматик на основі прихованих моделей Маркова;

3) побудувати модель стохастичних контекстно-вільних граматик на основі узагальнення прихованих моделей Маркова та дослідити властивості цієї моделі;

4) розробити програмну реалізацію моделей;

5) перевірити узгодженість моделей емпіричним шляхом.

Об'єктом дослідження є формальна модель природних мов.

Предметом дослідження є побудова поліпшеної моделі природної мови на основі стохастичної контекстно-вільної граматики в нормальній формі Грейбах.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи теорії формальних мов та граматик, теорії складності, теорії автоматів, теорії імовірності, математичної лінгвістики та машинного навчання.

Наукова новизна отриманих результатів полягає в вперше запропонованій формальній моделі мови на основі узагальнення прихованої моделі Маркова; в доведенні еквівалентності стохастичних контекстно-вільних граматик та прихованих моделей Маркова другого порядку за спостереженнями зі стеком.

Практичне значення результатів полягає у використанні поліпшеної моделі природних мов в методах криптоаналізу. Запропонована модель має меншу складність практичної реалізації у порівнянні з наявними сучасними моделями та краще алгоритмізується.

Апробація результатів та публікації. Результати були апробовані на конференціях:

1) Yevhen Hrubiian. New Probabilistic Model of Stochastic Context-Free Grammars in Greibach Normal Form // Інформаційні технології та комп'ютерне моделювання, Івано-Франківськ - 2017

2) Євген Грубіян. Еквівалентність стохастичних контекстно-вільних граматик та прихованих моделей Маркова зі стеком // Безпека інформації у інформаційно-телекомунікаційних системах, Буча - 2017

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

В цьому розділі викладено основні теоретичні результати і визначення формальної теорії мов, деякі визначення з теорії прихованих моделей Маркова та теорії обробки природних мов.

1.1 Природні мови

З часу виникнення перших природних мов пройшли десятки тисяч років, проте сьогодні важко уявити засіб комунікації більш дієвий ніж мова. Вивченням природних мов займається лінгвістика, що об'єднує безліч напрямків вивчення аспектів природних мов. Серед них особливо можна виділити граматику - науку про структуру мови. В свою чергу граматика поділяється на синтаксис і морфологію. Морфологія вивчає структуру окремих слів, що складаються з літер алфавіту, тоді як синтаксис вивчає правила, за якими слова складають граматично правильні речення. Вивченням змістовних та інформативних характеристик мови займається семантика. З різними поглядами на співвідношення граматики та семантики пов'язано чимало дискусій. В 50х роках американський лінгвіст та публіцист Ноам Хомський публікує праці з лінгвістики, що свого роду перевертають звичні уявлення про мову, вводячи поняття генеративної лінгвістики та генеративної граматики. За уявленнями Хомського всі мовленнєві структури мають над собою більш абстрактні і загальні структури, а правила за якими більш абстрактні структури породжують менш абстрактні і утворюють генеративну граматику. Тобто центральним місцем в лінгвістиці згідно Хомським є граматика, тоді як семантика мови в дечому визначається

граматикою та її глибинною структурою[4,5]. Цей підхід сьогодні в середовищі багатьох європейських лінгвістів визнаний не зовсім сучасним і некоректним, оскільки існує дуже багато контрприкладів в природних мовах, які не пояснюються або пояснюються слабо з точки зору теорій Хомського[22].

З розвитком сучасного штучного інтелекту тісно пов'язана проблема аналізу та синтезу природних мов. Яким чином машина, яка не має власної свідомості, а тільки запрограмований набір логічних команд може ефективно розрізняти правильні граматично і семантично фрази від не правильних? Яким чином вона може утворювати осмислені фрази та прогнозувати слова які може сказати співбесідник? Алан Тьюрінг в своїх роботах ввів тест, який повинен пройти справжній штучний інтелект. Цей тест включає позитивні відповіді на згадані вище питання, тому обробка природних мов є надзвичайно важливою гілкою computer science[8].

Теорія граматик, якій значною мірою є присвячена дана наукова робота, важлива не тільки при обробці природних мов. В біології теорія граматик застосовується при аналізі структур ланцюгів нуклеотидів в транспортній РНК[11]. В теорії компіляторів мов програмування при побудові синтаксичних парсерів. В криптоаналізі при побудові моделей відкритого тексту. В теорії розпізнавання образів при виділенні глибоких структур в зображеннях. І це далеко не повний перелік застосування теорії граматик.

1.2 Граматики, формальні мови та ієрархія Хомського

Сучасна формалізація поняття мови почалось з робіт Ноама Хомського в 50-60х роках. Його метою була повна формалізація англійської мови, перенесення її граматики і правил на мову логіки та

математики. Повної формалізації не досягнуто і до сьогодні, проте було запропоновано багато концепцій, які дуже важливі сьогодні. Наведемо формальні базові визначення з теорії формальних мов та граматик[1].

Нехай задано деякий алфавіт, тобто скінчену множину Σ.

Визначення 1.1. *Мовою L над алфавітом* Σ називається деяка підмножина $L \subset \Sigma^*$. Де Σ^* - замикання Кліні множини Σ, тобто всі можливі скінчені конкатенації елементів алфавіту Σ. Такі конкатенації, тобто ланцюжки послідовно записаних елементів алфавіту називаються *словами* або *реченнями*, в залежності від контексту.

Зауважимо що множина Σ^* нескінченна, окрім випадку коли алфавіт порожній. Наведемо приклад мови:

Приклад 1.1. Нехай наш алфавіт:

$$\Sigma = \{a, e, h, t\}$$

Тоді деяка мова L над цим алфавітом:

$$L = \{ae, the, ht, ttt, aaaa\}$$

Зауважимо, що в даному випадку, коли алфавіт складається з літер, елементи нашої мови L називають *словами*. Якщо елементи алфавіту - власне самі слова, то елементи мови прийнято називати *реченнями*(до прикладу в теорії синтаксису). Наприклад:

$$\Sigma = \{Bill, Mary, loves, \text{`` ''}\}$$

$$L = \{Bill\ loves\ Mary, Mary\ loves\ Bill\}$$

З поняттям мови тісно пов'язане поняття *граматики*. Неформально граматика - множина правил, що породжують мову. Для формального визначення граматики потрібні визначення термінальних та нетермінальних символів.

Визначення 1.2. *Термінальним символом або терміналом* називають елемент $x \in \Sigma$, де Σ - звичний нам алфавіт, з якого будуть складатись слова мови, які породжує граматика. Алфавіт Σ називають *алфавітом термінальних символів*. Часто в множину терміналів включають порожній символ ϵ.

Визначення 1.3. *Нетермінальним символом, нетерміналом або змінною* називають елемент $x \in N$, де N - алфавіт, що предствляє деякі категорії або абстракції мови, що не входять безпосередньо в алфавіт, з якого будуються слова мови. N називають *алфавітом нетермінальних символів*.

Визначення 1.4. *Граматикою* G називають впорядковану четвірку $G = \langle N, \Sigma, S, R \rangle$ Де N - алфавіт нетермінальних символів, Σ - алфавіт термінальних символів, S - стартовий нетермінальний символ, R - множина *продукцій* граматики, де продукціями називають правила заміни ланцюжків символів виду:

$$A \to B$$

Де A,$B \in (N \cup \Sigma)^*$

Визначення 1.5. Слово $w \in (N \cup \Sigma)^*$ називається *породженим* з слова $v \in (N \cup \Sigma)^*$ в граматиці G, якщо існує правило виводу $g \in G$, застосування якого до слова v продукує слово w. Позначається $v \Rightarrow w$. Схожим чином будем позначати $v \Rightarrow^* w$, якщо існує ланцюг правил виводу, що дозволяють з слова v отримати слово w, послідовно застосовуючи їх.

Граматики перетворюють стартовий нетермінал S згідно правил G, послідовно замінюючи ліві частини продукцій на праві. Наведемо приклад:

Приклад 1.2. Нехай дана граматика

$$G = \langle \{S\}, \{a, b, \epsilon\}, S, \{S \to aSa,\ S \to bSb,\ S \to a,\ S \to b,\ S \to \epsilon\} \rangle$$

Мова $L(G)$, породжувана цією граматикою складається з ланцюжків букв

a, b, які однаково читаються зправа на ліво і зліва направо, це так звані *паліндроми.*

З поняттям граматики тісно пов'язане поняття мови, що вона породжує.

Визначення 1.6. Кажуть, що граматика $G = \langle N, \Sigma, S, R \rangle$ породжує мову $L(G)$ тоді і тільки тоді коли $\forall x \in L(G)\ \exists\ S \Rightarrow^* x$

Н. Хомський класифікував всі мови за деякими ознаками строгості граматик які їх породжують:

Вільні мови

Контекстно-залежні мови

Контекстно-вільні мови

Регулярні мови

Регулярні мови - найбільш вузький клас мов в ієрархії. Вони розпізнаються детермінованими скінченими автоматами, тому вони стали дуже поширеними в мовах програмування для написання регулярних виразів - шаблонів пошуку і допускання рядків.

Контекстно-вільні мови - один із найцікавіших класів мов, що достатньо добре формалізують природні мови, ланцюжки нуклеотидів в ДНК, мови програмування та багато іншого. Дана наукова робота присвячена вивченню аспектів контекстно-вільних мов та граматик, тому далі зупинимось детальніше на них.

Визначення 1.7. *Контекстно-вільною граматикою* G називається граматика $G = \langle N, \Sigma, S, R \rangle$, всі правила виводу R якої мають вигляд:

$$A \to \gamma,\ A \in N,\ \gamma \in (N \cup \Sigma)^*$$

Відповідно *контекстно-вільною мовою* називається мова, що породжена деякою контекстно-вільною граматикою.

Зауважимо, що назва контекстно-вільної граматики дещо оманлива і аж ніяк не вказує на те, що слова не залежать від контексту, в яких вони

стоять в реченні, а лише на той факт, що граматичний вивід не залежить від контексту, який оточує нетермінал. Н. Хомський запропонував КВ-граматики для опису граматики англійської. До кінця це не вдалось нікому, проте моделі англійської, як КВ-мови виявились досить непоганими. Виявилось, що не тільки англійська може бути добре описана КВ-граматикою, але і багато інших природних мов, проте далеко не всі, оскільки відомо декілька природних мов, що точно не є контекстно-вільними. В лінгвістиці речення зазвичай представляють у вигляді двох груп слів - групи підмета(*noun phrase*) та групи присудка(*verb phrase*). Формальний граматичний еквівалент: $S \rightarrow NP\ VP$. Варто зазначити, що клас контекстно-вільних мов досить широкий і еквівалентний недетермінованим автоматам з магазинною пам'яттю(стеком)[1]. Це дозволяє на практиці ефективно створювати синтаксичні парсери контекстно-вільних мов.

Контекстно-залежні мови - мови дещо більш складні ніж контекстно-вільні, оскільки граматики що їх породжують, враховують термінальні та нетермінальні символи, що оточують нетермінал в правилах виводу, таким чином контекст впливає на граматичні породження. Одним із найцікавіших прикладів серед природних мов є швейцарський діалект німецької, де речення *... das mer em Hans es huus hälfed aastriiche*, що перекладається *... що ми допомогли Гансу розмалювати будинок* є прикладом перехресної залежності між іменниками та дієсловами. Дослівний переклад на українську - *... що ми Гансу будинок допомогли розмалювати.* Саме ці синтаксично-розірвані групи підмета та присудка і створюють контекстну залежність в граматиці.

Вільні мови - мови, що породжені довільною граматикою. В літературі по формальній теорії алгоритмів цей клас також називають *рекурсивно-перечислюваними мовами.* Еквівалент - машина Т'юрінга.

1.3 Нормальні форми контекстно-вільних граматик

Для зручності представлення, парсингу мов та доведення теорем контекстно-вільні граматики часто представляють у вигляді, де правила виводу мають певний специфічний вигляд.

Визначення 1.8. Граматика знаходиться у *нормальній формі Хомського (CNF)*, якщо всі правила виводу мають вигляд:

$$A \to a$$

$$A \to A_1 A_2$$

Де $a \in \Sigma$, $A, A_1, A_2 \in N$

Теорема 1.1. *Кожна граматика у нормальній формі Хомського є контекстно-вільною і кожну контекстно-вільну граматику можна привести до нормальної форми Хомського[1].*

Визначення 1.9. Граматика знаходиться у *нормальній формі Грейбах (GNF)*, якщо всі правила виводу мають вигляд:

$$A \to aA_1A_2 \dots A_n$$

$$S \to \epsilon$$

Де $a \in \Sigma$, $A \in N$, $A_1, A_2 \in N \setminus \{S\}, n \in \{0,1, \dots\}$

Теорема 1.2. *Кожна граматика у нормальній формі Грейбах є контекстно-вільною і кожну контекстно-вільну граматику можна привести до нормальної форми Грейбах[1, 7].*

В нашій роботі будемо використовувати один із варіантів нормальної форми Грейбах, а саме *2-нормальну форму Грейбах.*

Визначення 1.10. Граматика знаходиться у *2-нормальній формі*

Грейбах (2-GNF), якщо всі правила виводу мають вигляд:

$$A \to aA_1A_2$$

$$A \to aA_1$$

$$A \to a$$

$$S \to \epsilon$$

Де $a \in \Sigma$, $A \in N$, $A_1, A_2 \in N \setminus \{S\}$

Теорема 1.3. *Кожна граматика у 2-нормальній формі Грейбах є контекстно-вільною і кожну контекстно-вільну граматику можна привести до 2-нормальної форми Грейбах[16, 17].*

Наступна властивість нормальної форми Грейбах буде в подальшому дуже корисною для побудови моделей

Твердження 1.1. *Для породження речення довжини L в граматиці, що знаходиться в нормальній формі Грейбах потрібно рівно L застосувань правил виводу.*

Доведення. Всі правила виводу граматики в нормальній формі Грейбах мають вигляд: $A \to a\gamma$, де γ - довільна послідовність нетерміналів окрім стартового символу. Таким чином стан граматичного виводу wA(w - ланцюжок терміналів, що були виведені граматикою раніше) після застосування правила стане $wa\gamma$, кількість терміналів в стані збільшиться на одиницю. Таким чином, застосовуючи правила виводу n разів в лівій частині стану граматичного виводу буде знаходитись n терміналів. □

Ця властивість дозволила свого часу суттєво розвити теорію синтаксичних парсерів та компіляторів мов програмування, оскільки для парсингу речень, що належать мові яка породжена граматикою в нормальній формі Грейбах top-down парсеру необхідна кількість кроків, що співпадає з довжиною речення.

Вибір саме 2-нормальної форми Грейбах для подальшого дослідження обумовленений обмеженням кількості нетерміналів в правих частинах правил виводу, що робити цю нормальну форму зручною для подальшого дослідження. Також існують поліноміальні алгоритми зведення КВ-граматики до нормальних форм Хомського та Грейбах, але в даному розділі приводити їх не будемо[1, 16, 17].

1.4 Елементи тензорного числення

Далі нам у роботі будуть потрібні дещо специфічні об'єкти, що в нашому застосуванні будуть нагадувати багатовимірні матриці.

Визначення 1.11. *Тензором* рангу (n,m) над векторним простором V будемо називати елемент тензороного добутку:

$$V^{\otimes n} \otimes V^{*\otimes m}$$

Де V^* - спряжений простір до V(простір всіх лінійних функціоналів над V). $V^{\otimes n}$ - тензорний добуток V на самого себе n раз.

До прикладку тензори рангу (1,0) - звичайні вектори із V, тензори рангу (1,1) - лінійні перетворення простору V(матриці). Тензори рангу (2,1) та (1,2) подібні до трьох-вимірних матриць. Вони часто будуть застосовуватись в нашій роботі.

Часто рангом тензора називають суму $n + m$. Далі будемо користуватися саме такою нотацією для рангу.

1.5 Класичні моделі природних мов

Відомо що природні мови дуже не рівноімовірні, ентропія(кількість інформації на символ тексту) природних мов набагато менша за ентропію рівноімовірного ансамблю, тому існує багато статистичних властивостей мов, які досить добре прослідковуються в межах тематичних текстів. Клод Шенон в своїх роботах ввів декілька моделей природних мов, що знайшли широке застосування в криптографії, оскільки вони досить добре описують базові статистичні властивості мови, що важливо при криптоаналізі. Розглянемо моделі, що були ним запропоновані:

– Модель *M0* - текст представляє послідовність незалежних випадкових символів, що мають рівноімовірний розподіл на алфавіті: $P(x_k = i) = 1/n$, n - розмір алфавіту. Такий текст буде мати максимальну ентропію на символ тексту.

– Модель *M1* - текст представляє собою послідовність випадкових символів алфавіту, імовірності яких збігаються з оригінальними їх імовірностями появи у мові. Тобто $P(x_i = a) = p_a$, де x_i - і-тий символ тексту p_a - оцінка імовірності появи символу a в тексті, отримана статистичним шляхом.

– Модель *M2* - текст представляє собою послідовність випадкових біграм, тобто пар символів алфавіту, розподіл яких збігається з оригінальним розподілом біграм у мові $P(x_i = ab) = p_{ab}$, де x_i - і-та пара символів у тексті(пари не перетинаються), p_{ab} - оцінка імовірності біграми ab, отримана статистичним шляхом.

– Модель *M3* - текст представляється у вигляді ланцюга Маркова, тобто

$$P(x_k|x_{1:k-1}) = P(x_k|x_{k-1}),\ P(x_k = a|x_{k-1} = b) = p_{ab}$$

Дані моделі описують мову з дуже простого статистичного боку, враховуючи звичайні неоднорідності вживання різних букв в мові і як

правило вони не підходять для більш точного опису властивостей мови. Такі ж моделі можна побудувати, заміняючи букви на слова, проте оскільки кількість слів у мові набагато більша ніж букв, очікувати осмислених текстів не варто.

Одним із способів поєднати формальну теорію мов і граматик та природні статистичні властивості мов є введення стохастичних контекстно-вільних граматик, що враховують як граматичні властивості мови так і деякі семантичні, що продиктовані природніми статистичними особливостями.

1.6 Стохастичні контекстно-вільні граматики

Дамо формальне визначення стохастичних контекстно-вільних граматик.

Визначення 1.12. *Стохастичною(імовірнісною) контекстно-вільною граматикою* G називають впорядковану п'ятірку $G = \langle N, \Sigma, S, R, \mathbb{P} \rangle$ Де $G' = \langle N, \Sigma, S, R \rangle$ - контекстно-вільна граматика, $N = \{A_1, \ldots, A_m\}$, $S = A_1$, а $\mathbb{P} = \{\mathbb{P}_1, \ldots, \mathbb{P}_m\}$ - сімейство імовірнісних розподілів на G, таке що на кожній множині продукцій, що починаються з одинакового нетерміналу:

$$G_i = \{g \in G | g \colon A_i \to \gamma_i,\ \gamma_i \in (N \cup \Sigma)\}$$

задано імовірнісний розподіл $\mathbb{P}_i$, $i = \overline{1,\ m}$

Також введемо поняття *стохастичної мови*, що взагалі кажучи може бути породжена будь якою стохастичною граматикою, а не тільки контекстно-вільними.

Визначення 1.13. Кажуть, що стохастична граматика $G = \langle N, \Sigma, S, R, \mathbb{P} \rangle$ породжує стохастичну мову $L(G)$ тоді і тільки тоді

коли $\forall x \in L(G)\colon S \Rightarrow^* x$ та на $L(G)$ задано імовірнісний розподіл, індукований граматикою G: $\forall x \in L(G)\colon P(x) = P(S \Rightarrow^* x)$

Стохастичні контекстно-вільні граматики є досить потужним засобом для моделювання досить різних за природою структур, не враховуючи природні мови це можуть бути ланцюжки ДНК, певні структури в зображеннях, природні фрактали, музичні твори та інше. Варто зазначити що багато результатів в теорії стохастичних граматик були отримані в результаті дослідження саме первинних і вторинних структур транспотрної РНК, в тому числі ідея застосування прихованих моделей Маркова, яку в подальшому будемо розвивати в роботі походить з біоінформатики[11,14].

Для нас особливий інтерес будуть складати контекстно-вільні граматики в 2-нормальній формі Грейбах.

1.7 Приховані моделі Маркова

Для подальшого дослідження також знадобиться апарат прихованих моделей Маркова, тому опишемо їх та окреслимо основні проблеми та способи їх вирішення. Отже, як відомо ми називаємо стохастичний процес $\{X_t,\, t \in \mathbb{N}\}$ *ланцюгом Маркова* з множиною станів E, якщо виконується $\forall t > 1\colon P(X_t = k|X_{1:t-1}) = P(X_t = k|X_{t-1})$, $k \in E$. Позначення $X_{1:t-1}$ трактується як вектор $(X_1, \ldots, X_{t-1})$. Одна із найпоширеніших варіацій - однорідний ланцюг Маркова однозначно задається матрицею переходів між станами $\mathbb{P}$, яка не змінюється для будь яких $n \in \mathbb{N}$ та стартовим розподілом π_0. Із вигляду матриці $\mathbb{P}$ можна вивести багато властивостей ланцюга, таких як періодичність, рекурентність та ергодичність. Ланцюги Маркова є ймовірнісними аналогами систем, що задані деякими задачами Коші, тобто диференційними рівняннями разом з початковими умовами,

тому вони і стали такими широко розповсюдженими в сучасній математиці, фізиці та економіці. Одним із найвдаліших застосувань концепції ланцюгів Маркова є приховані моделі Маркова, що виникли з потреб розпізнавання мови на фонетичному рівні та інших проблем, що пов'язані з рядом спостережень, що безпосередньо були наслідком деякого іншого прихованого процесу, який не можна спостерігати. Дамо спочатку визначення прихованої моделі Маркова і наведемо приклади.

Визначення 1.14. *Прихованою моделлю Маркова* з простором латентних(прихованих) станів $E = \{1, \ldots, n\}$ та простором спостережень $O = \{1, \ldots, m\}$ називається двохвимірний стохастичний процес $\{(X_t, Y_t),\, t \in \mathbb{N}\}$, $X_t \in E$, $Y_t \in O$, що має властивості:

– $\{X_t,\, t \in \mathbb{N}\}$ - однорідний ланцюг Маркова.

– $\forall t \in \mathbb{N}$: $P(Y_t = k | X_{1:t}) = P(Y_t = k | X_t)$, $k \in O$

Тобто неформально, прихована марківська модель - це деякий процес, що залежить від станів ланцюга Маркова, які безпосередньо не можна спостерігати, але можна спостерігати прояви цього ланцюга. Прихована модель Маркова описується двома матрицями - матрицею переходів між латентними станами:

$$\mathbb{P} = \{p_{ij}\}_{i,j=1..n},\; p_{ij} = P(X_t = j | X_{t-1} = i)$$

та матрицею спостережень:

$$\mathbb{T} = \{q_{ij}\}_{i=1..n,j=1..m},\; p_{ij} = P(Y_t = j | X_t = i)$$

Приклад 1.3. Нехає є деякий фонетичний розпізнавач, який на вхід приймає деякий сигнал, що представляє собою послідовність звуків які вимовляє людина. Тоді відносно розпізнавача сигнал може бути уявлений як вихід деякої прихованої моделі Маркова, що в якості прихованого ланцюга має природну послідовність букв в мові. Таким чином, щоб ідентифікувати послідовність букв $X_{1:t}$ за звуками $y_{1:t}$, що ми спостерігаємо достатньо максимізувати імовірність: $P(X_{1:t} | Y_{1:t} = y_{1:t})$,

дана задача вирішується за допомогою відомого алгоритму Вітербі(*Viterbi algorithm*)[13].

Крім вище описаної задачі в рамках теорії прихованих моделей Маркова вирішується задача знаходження найбільш імовірного ланцюжка спостережень $y_{1:t}$: $P(Y_{1:t} = y_{1:t}) \to max$ за допомогою алгоритму прямого ходу(*Forward algorithm*) та задача ідентифікації параметрів(матриці $\mathbb{P}$, $\mathbb{T}$) прихованої марківської моделі(задача навчання) за допомогою алгоритму Баума-Велша(*Baum-Welch algorithm*)[9,10] з складністю $\mathcal{O}(nm^2)$, що базується на алгоритмі прямого-зворотнього ходу(*Forward-Backward algorithm*)

Далі нам знадобиться деяке узагальнення прихованої моделі Маркова, а саме прихована модель Маркова порядку r за спостереженнями.

Визначення 1.15. *Прихованою моделлю Маркова порядку* r з простором латентних(прихованих) станів $E = \{1, \ldots, n\}$ та простором спостережень $O = \{1, \ldots, m\}$ називається двовимірний стохастичний процес $\{(X_t, Y_t),\, t \in \mathbb{N}\}$, $X_t \in E$, $Y_t \in O$, що має властивості:

– $\{X_t,\, t \in \mathbb{N}\}$ - однорідний ланцюг Маркова.

– $\forall t \in \mathbb{N}$: $P(Y_t = k | X_{1:t}) = P(Y_t = k | X_{t-r-1:t})$, $k \in O$

1.8 Огляд сучасних підходів до навчання стохастичних контекстно-вільних граматик

Оглянемо деякі сучасні підходи для навчання стохастичних контекстно-вільних граматик. Найпоширенішим підходом є навчання стохастичної контекстно-вільної граматики в нормальній формі Хомського за допомогою алгоритму *Inside-Outside*[2,3]. Алгоритм в чомусь подібний до алгоритму *Forward-Backward*, проте він має більшу складність $\mathcal{O}(n^3 m^3)$, де n, m - відповідно потужності алфавітів

нетермінальних і термінальних символів. Оскільки при великих значеннях n і m складність алгоритму може бути досить значною існує проблема розробки ефективніших методів.

Висновки до розділу 1

В розділі було розглянуто проблематику формалізації природних мов, теоретичні результати формальної теорії мов, граматик, теорії прихованих моделей Маркова та їхнє практичне застосування. Зокрема було розглянуто класичні підходи до навчання стохастичних контекстно-вільних граматик та класичні підходи до побудови моделей відкритого тексту.

2 ПОБУДОВА НОВОЇ МОДЕЛІ СТОХАСТИЧНОЇ КОНТЕКСТНО-ВІЛЬНОЇ ГРАМАТИКИ

Метою цього розділу є побудова нової моделі стохастичної контекстно-вільної граматики, що дозволить застосовувати її для моделювання відритих текстів у криптоаналізі, також побудована модель дозволить розробку ефективніших алгоритмів навчання стохастичної контекстно-вільної граматики. Для досягнення мети розглянемо побудову моделі для стохастичних регулярних граматик, а потім перейдемо до складнішого випадку контекстно-вільних граматик.

2.1 Модель стохастичних регулярних граматик

Розглянемо побудову моделі стохастичних регуляриних граматик, що за своєю структурою є простішими ніж контекстно-вільні. Нагадаємо, що регулярні граматики описується правилами виводу виду:

$$A \to aB$$

$$A \to a$$

де A,B - нетермінали, a - термінал.

Регулярні граматики, як відомо еквівалентні скінченим автоматам, тому можна провести деяку аналогію з прихованими моделями Маркова, ототожнивши простір прихованих станів з алфавітом нетермінальних символів, а простір спостережень з алфавітом термінальних символів, але виникає деяка проблема з описом імовірностей спостережень. Нехай

деяка граматика містить правила:

$$(p_1)\ A \to aA_1$$

$$(p_2)\ B \to bA_1$$

Ми ввели префіксну нотацію для імовірностей переходів. До прикладу запис $(0.5)\ A \to aA_1$ означає що імовірність застосування правила $A \to aA_1$ дорівнює 0.5. В класичній прихованій марківській моделі спершу визначаєть наступний латентний стан, а потім відповідно до нього визначається наступне спостереження, а у випадку регулярної граматики застосування правила передбачує одночасність переходу і спостереження. Тому в записаному вище прикладі граматики у випадку ототожнення стохастичної регулярної граматики і прихованої моделі Маркова із станів A, B переходимо до стану A_1, а потім виникає проблема який із термінальних символів нам згенерувати: a чи b. Щоб уникнути цієї проблеми введемо *приховану модель Маркова другого порядку за спостереженнями*:

Визначення 2.1. *Прихованою моделлю Маркова другого порядку за спостереженнями* з простором латентних(прихованих) станів $E = \{1,\ldots,n\}$ та простором спостережень $O = \{1,\ldots,m\}$ називається двохвимірний стохастичний процес $\{(X_t, Y_t),\ t \in \mathbb{N}\},\ X_t \in E,\ Y_t \in O$, що має властивості:

– $\{X_t,\ t \in \mathbb{N}\}$ - однорідний ланцюг Маркова.

– $\forall t > 1$: $P(Y_t = k|X_{1:t}) = P(Y_t = k|X_t, X_{t-1}),\ k \in O$

Зауважимо, що як тільки буде застосоване правило виду $A \to a$, то граматичний вивід закінчується і описана вище прихована модель Маркова переходитиме до порожнього стану ϵ із завершенням своєї роботи. Замість матриці спостережень в випадку розглянутого узагальнення буде використаний тензор 3-го рангу. Позначати параметри прихованої модель Маркова другого порядку за спостереженнями будемо впорядкованою трійкою: $\lambda = \langle Q, T, \pi \rangle$, де Q - матриця переходів, T -

тензор спостережень, π - розподіл початкового латентного стану. Таким чином ввівши таке узагальнення можемо сформулювати і довести наступну теорему:

Теорема 2.1. *Для кожної стохастичної регулярної граматики існує прихована модель Маркова другого порядку за спостереженнями, що допускає таку саму стохастичну мову.*

Доведення. Доведення будемо проводити конструктивним шляхом, показуючи для кожного етапу граматичного виводу його еквівалент в термінах прихованих моделей Маркова. Нехай маємо стохастичну регулярну граматику $G = \langle N, \Sigma, S, R, \mathbb{P} \rangle$. $|N| = n$, $|\Sigma| = m$.

– Введемо приховану модель Маркова другого порядку за спостереженнями $\{(X_t, Y_t)\}$ з параметрами $\lambda = \langle Q, T, \pi \rangle$ з простором латентних станів $N \cup \{\epsilon\}$ та простором спостережень Σ.

– Початковий розподіл π задамо у вигляді: $\pi \colon P(X_1 = S) = 1$, тобто модель стартує з початкового нетерміналу граматики.

– Ініціалізуємо матрицю переходів Q наступним чином:

$$q_{AB} = \sum_{a \in \Sigma} P(A \to aB) \; \forall A \in N, \, B \in N \cup \{\epsilon\}$$

– Ініціалізуємо тензор спостережень T наступним чином:

$$t_{ABa} = \frac{P(A \to aB)}{q_{AB}}$$

Таким чином, якщо запустити модель, вона буде в точності імітувати вивід слова в стохастичній регулярній граматиці. Термінація моделі буде здійснена при досяганні порожнього стану ϵ, тобто в термінах граматики - застосування правила виду $A \to a$. □

Твердження 2.1. *Алгоритм перетворення стохастичної регулярної граматики до прихованої моделі Маркова другого порядку за спостереженнями має складність $\mathcal{O}(l)$, де l - кількість правил виводу в граматиці.*

Доведення. З вигляду описаного вище нами алгоритму легко побачити, що для кожного правила граматики необхідно оновлювати тензор спостережень та матрицю переходів. Для цього необхідно $\mathcal{O}(l)$ операцій. Зауважимо що при ініціалізації матриці переходів не потрібно на кожному кроці сумувати по всім терміналам. □

Оскільки був доведений факт еквівалентності стохастичних регулярних граматик і прихованих моделей Маркова другого порядку за спостереженнями, можемо використати модифікований алгоритм Баума-Велша для навчання за множиною ланцюжків спостережень[15]. Варто зазначити, що факт еквівалентності був відомий і раніше, зокрема в [9] описано механізм еквівалентності, проте формально не описано умови, які накладені при цьому на приховану модель Маркова(її другий порядок).

2.2 Ланцюги Маркова зі стеком

Тепер можна узагальнити модель побудовану для стохастичних регулярних граматик на випадок стохастичних контекстно-вільних граматик. Для цього використаємо той факт, що кожна контекстно-вільна граматика може бути представлена у вигляді граматики в 2-нормальній формі Грейбах, оскільки структура її дуже подібна до регулярної граматики з єдиним обмеженням, що в правій частині правил виводу може знаходитись максимально 2 нетермінали. Така структура правил виводу наштовхує на думку, що для послідовного застосування правил виводу нам потрібно деяка пам'ять. Дійсно, в теорії граматик доведено факт, що стохастичні контекстно-вільні граматики еквівалентні недетермінованим автоматам з магазинною пам'яттю(стеком) саме через представлення граматики в нормальній формі Грейбах.

Введемо деяке узагальнення ланцюга Маркова, а саме - *ланцюг Маркова зі стеком*:

Визначення 2.2. *Стеком над множиною* X називається скінчена множина S, що складається з елементів множини X з введеним на ній лінійним порядком. Максимальний елемент відносно цього порядку будемо називати вершиною стеку.

Введемо деякі операції зі стеком S:

1) $S.Top$ - повертає вершину стеку, якщо він не порожній, інакше порожній символ ϵ.

2) $S.Push(a)$ - записує в стек нову вершину a, якщо вона не дорівнює порожньому символу ϵ.

3) $S.Pop$ - вилучає стару вершину з стеку.

Визначення 2.3. *Ланцюгом Маркова зі стеком* з простором станів E називається стохастичний процес $\{(X_t, S_t)\ t \in \mathbb{N}\}$ ($X_t \in E \cup \{\epsilon\}$, S_t - стек над множиною $E \cup \{\epsilon\}$ в момент часу t), такий що переходи між станами здійснюються за алгоритмом:

1) $S_t = S_{t-1}$

2) $P(X_t = i,\ S_t.Push(k)|X_{1:t-1}) = P(X_t = i,\ S_t.Push(k)|X_{t-1})$, $P(X_t = i,\ S_t.Push(k)|X_{t-1} = j) = p_{jik}$

3) Якщо після переходу $X_t = \epsilon$ та $S_t.Top \neq \epsilon$, то новий стан X_t визначається: $X_t = S_t.Top$, $S_t.Pop$. Якщо $X_t = \epsilon$ та $S_t.Top = \epsilon$, то ланцюг завершується.

Ланцюг Маркова зі стеком визначається тензором рангу 3 перехідних імовірностей $T = \{p_{jik}\}_{i,j,k \in E \cup \{\epsilon\}}$ та початковим розподілом π: $P(X_1 = i) = \pi_i$. Стек ініціалізується порожнім, хоча можливе узагальнення на довільний початковий стек.

Наведемо приклад ланцюга Маркова зі стеком.

Приклад 2.1. $E = \{1, 2, 3\}$, $\pi_1 = 1$

$$T[1] = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0 & 0 & 0 \end{pmatrix}, T[2] = \begin{pmatrix} 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0 & 0 & 0 \end{pmatrix}$$

$$T[3] = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0 & 0.1 \\ 0.1 & 0 & 0 & 0.2 \end{pmatrix}$$

Приклад згенерованого ланцюга: $(1, \varnothing), (2, \{1\}), (1, \varnothing), (3, \varnothing), (\epsilon, \varnothing)$

Концепція ланцюга Маркова зі стеком є фундаментом для побудови нової моделі стохастичної контекстно-вільної граматики. Дійсно, робота ланцюга дуже схожа на нетермінальну частину граматичного виводу в граматиці, що знаходиться в 2-нормальній формі Грейбах.

2.3 Прихована модель Маркова зі стеком

Отже, було показано еквівалентність стохастичних регулярних граматик і прихованих моделей Маркова другого порядку за спостереженнями. Тепер можна узагальнити випадок регулярної граматики на граматики в 2-нормальній формі Грейбах. Для цього скористаємось концепцією ланцюгів Маркова з стеком та прихованими

моделями Маркова другого порядку, синтезуючи їх в одне ціле.

В попередньому розділі було помічено схожість граматичного виводу в стохастичних граматиках, що знаходяться в 2-нормальній формі Грейбах і власне введених раніше ланцюгів Маркова зі стеком. Це наштовхує на думку застосувати підхід ланцюгів Маркова зі стеком до стохастичних граматик в 2-нормальній формі Грейбах. Застосовуючи концепцію прихованих моделей Маркова можна дійти висновку, що стохастичні граматики в 2-нормальній формі Грейбах еквівалентні *прихованим моделям Маркова другого порядку за спостереженнями зі стеком*. Далі введемо відповідні означення та доведемо факт даної еквівалентності.

Дійсно, всі правила виводу в 2-нормальній формі Грейбах мають вигляд:

$$A \to a$$

$$A \to aA_1$$

$$A \to aA_1A_2$$

Застосування першого виду правил буде приводити до переходу в стан, що знаходиться на вершині стеку з подальшим виштовхуванням вершини стеку. Застосування другого виду правил буде аналогічне функціонуванню регулярної граматики(прихованої моделі Маркова другого порядку за спостереженнями), тобто відбувається перехід до наступного стану A_1 без дій над стеком. Застосування же третього виду правил передбачає перехід до наступного стану A_1 з подальшим заштовхуванням в стек стану A_2. Термінація моделі буде здійснена після застосування правила першого виду при порожньому стеці. Введемо визначення вище згаданої *прихованої моделі Маркова другого порядку за спостереженнями зі стеком*, що дозволить моделювати стохастичні граматики в 2-нормальній формі Грейбах.

Визначення 2.4. *Прихованою моделлю Маркова другого порядку за*

спостереженнями зі стеком з простором латентних(прихованих) станів $E = \{1, \ldots, n\}$, простором спостережень $O = \{1, \ldots, m\}$ та стеком називається стохастичний процес $\{(X_t, S_t, Y_t),\, t \in \mathbb{N}\}$, $X_t \in E$, $Y_t \in O$, S_t - стек над множиною E, переходи в якому здійснюються за алгоритмом:

1) Ініціалізація стеку: $S_t = S_{t-1}$

2) Перехід між латентними станами:

$$P(X_t = i,\, S_t.Push(k)|X_{1:t-1}) = P(X_t = i,\, S_t.Push(k)|X_{t-1}),$$
$$P(X_t = i,\, S_t.Push(k)|X_{t-1} = j) = p_{jik}$$

3) Спостереження:

$$P(Y_t = i|X_{1:t}, S_t.Top) = P(Y_t = i|X_t, X_{t-1}, S_t.Top),$$
$$P(Y_t = i|X_t = j, X_{t-1} = k, S_t.Top = l) = q_{lkji}$$

4) Якщо після переходу $X_t = \epsilon$ та $S_t.Top \neq \epsilon$, то новий стан X_t визначається: $X_t = S_t.Top$, $S_t.Pop$. Якщо $X_t = \epsilon$ та $S_t.Top = \epsilon$, то ланцюг завершується.

Прихована модель Маркова зі стеком визначається тензором рангу 3 перехідних імовірностей $T = \{p_{jik}\}_{i,j,k \in E \cup \{\epsilon\}}$, тензором рангу 4 імовірностей спостереження $Q = \{q_{lkji}\}_{lkj \in E \cup \{\epsilon\}, i \in O}$ та початковим розподілом $\pi\colon P(X_1 = i) = \pi_i$, $i \in E$. Стек ініціалізується порожнім, хоча можливе узагальнення на довільний початковий стек. Отже, сформулюємо наступну теорему:

Теорема 2.2. *Для кожної стохастичної контекстно-вільної граматики існує прихована модель Маркова другого порядку за спостереженнями зі стеком, що допускає таку саму стохастичну мову.*

Доведення. Доведення так само побудуємо конструктивним шляхом, показуючи еквівалентність граматичного виводу в обох моделях. Спочатку згадаємо, що кожну контекстно-вільну граматику можна

представити у вигляді граматики в 2-нормальній формі Грейбах: $G = \langle N, \Sigma, S, R, \mathbb{P} \rangle$, $|N| = n$, $|\Sigma| = m$.

– Введемо приховану модель Маркова другого порядку за спостереженнями зі стеком $\{(X_t, Y_t, S_t)\}$ з параметрами $\lambda = \langle T, Q, \pi \rangle$ з простором латентних станів $N \cup \{\epsilon\}$ та простором спостережень Σ.

– Початковий розподіл π задамо у вигляді: $\pi\colon P(X_1 = S) = 1$, тобто модель стартує з початкового нетерміналу граматики.

– Ініціалізуємо тензор переходів T наступним чином:

$$p_{ABC} = \sum_{a \in \Sigma} P(A \to aBC) \; \forall A \in N,\; B, C \in N \cup \{\epsilon\}$$

– Ініціалізуємо тензор спостережень Q наступним чином:

$$q_{CABa} = \frac{P(A \to aBC)}{p_{ABC}}$$

Таким чином, перехід моделі від стану (X_t, Y_t, S_t) до стану $(X_{t+1}, Y_{t+1}, S_{t+1})$ буде описувати один крок граматичного виводу. Термінація моделі буде здійснена при одночасному досяганні $X_t = \epsilon$ і $S_t = \varnothing$. □

Твердження 2.2. *Алгоритм перетворення стохастичної граматики в 2-нормальній формі Грейбах до прихованої моделі Маркова другого порядку за спостереженнями зі стеком має складність* $\mathcal{O}(l)$*, де* l *- кількість правил виводу в граматиці.*

Доведення. З вигляду описаного вище нами алгоритму легко побачити, що для кожного правила граматики необхідно оновлювати тензори спостережень та переходів. Для цього необхідно $\mathcal{O}(l)$ операцій. Зауважимо що при ініціалізації тензору переходів не потрібно на кожному кроці сумувати по всім терміналам. Реалізація алгоритму наведена в *додатку А*(файл hmm2s.cpp). □

Варто зазначити, що доведена теорема корелює з теоремою про еквівалентність автоматів з магазинною пам'яттю і контекстно-вільних граматик і є по суті її ймовірнісним аналогом, оскільки прихована модель

Маркова зі стеком за структурою нагадує стохастичний автомат з магазинною пам'яттю. В ході роботи були деякі вільності в трактуванні таких термінів, як термінація прихованої моделі Маркова. Строго кажучи термінацію можна визначити як потрапляння в порожній поглинаючий стан ϵ при порожньому стеці, тобто система залишиться в ньому вічно після потрапляння: $P(X_t = \epsilon | X_{t-1} = \epsilon, S_t.Top = \epsilon) = 1$.

Запропонована модель стохастичної контекстно-вільної граматики є по суті новою моделлю природної мови, узгодженість якої з реальними стохастичними контекстно-вільними граматиками буде перевірено в наступному розділі.

Висновки до розділу 2

В розділі було розроблено нову модель стохастичних контекстно-вільних граматик. Було доведено теорему про еквівалентність стохастичних регулярних граматик і прихованих моделей Маркова другого порядку та теорему про еквівалентність стохастичних контекстно-вільних граматик та прихованих моделей Маркова другого порядку зі стеком з поліноміальним алгоритмом зведення. Таким чином була запропонована нова модель природної мови на основі прихованих моделей Маркова другого порядку за спостереженнями зі стеком.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ЗАПРОПОНОВАНОЇ МОДЕЛІ

3.1 Сучасні моделі природних мов

В першому розділі нами було розглянуто класичні моделі природних мов: M0, M1, M2, M3. Розглянемо деякі сучасні підходи до моделювання природних мов.

Більш узагальненим і складним варіантом описаних моделей M0, M1, M2, M3 є n-грамна модель мови(текст утворює ланцюг Маркова порядку $n-1$), коли на появу слова впливають n-1 попередніх слів, задаючи відповідні розподіли:

$$P(x_k|x_{1:k-1}) = P(x_k|x_{k-n:k-1}),\ k > n$$

З статистичного боку n-грамна модель є досить хорошою, проте вона має дуже велику хибу, а саме те що кількість текстів для якісного навчання цієї моделі є дуже великим і експоненційно росте з ростом порядку моделі, але ріст порядку теж має свої недоліки при недостатньому об'ємі тексту для навчання. Тому n-грамні моделі згладжують за Лапласом. До прикладу оцінка максимальної правдоподібності для імовірності появи символа y при попередніх $x_{1:n-1}$ в n-грамній моделі[20]:

$$\hat{P}(X_k = y|X_{k-n:k-1} = x_{1:n-1}) = \frac{Cnt(x_{1:n-1},y)}{Cnt(x_{1:n-1})}$$

Згладжена за Лапласом оцінка:

$$\hat{P}(X_k = y|X_{k-n:k-1} = x_{1:n-1}) = \frac{Cnt(x_{1:n-1},y) + c}{Cnt(x_{1:n-1}) + cm^{n-1}}$$

Де $Cnt(x_{1:n-1},y)$ - кількість n-грами $(x_{1:n-1},y)$ в тексті, c - невелика константа згладжування, часто 0.5 або $1/m^{n-1}$, m - розмір алфавіту. Як було сказано друга оцінка краще у випадку коли розмір тексту для

навчання відносно невеликий.

Розгялнемо також деякі сучасні підходи до побудови моделей мови. Зокрема була запропонована так звана позиційна модель мови[19]. Її суть полягає в тому, що окрім власне слів у тексті враховується їх абсолютна позиція. Статистична оцінка імовірності деякого слова w на і-му місці буде мати вигляд:

$$\hat{P}(X_i = w) = \frac{c'(w,i)}{\sum_{w' \in V} c'(w',i)}$$

Де $c'(w,i) = \sum_{j=1}^{N} c(w,j)k(i,j)$. V - словник, $c(w,j)$ - індикатор того, що слово w стоїть на j-му місці(1 - якщо так, інакше 0), $k(i,j)$ - деяка вагова функція, не зростаюча відносно $|i-j|$ поширення слова на j позиції на сусідні з ним. Інтуїтивно можна розуміти, що слово вжите в деякому контексті буде збільшувати імовірність появи цього ж слова у цьому контексті.

Ще одним підходом до моделювання мови є застосування нейронних мереж для моделювання семантичної структури, зокрема такий підхід використовується в компанії *Google* для створення систем перекладу та розпізнавання мови[21].

Далі буде перевірена узгодженість запропонованої моделі природної мови із стохастичними контекстно-вільними граматиками.

3.2 Узгодженість запропонованої моделі з стохастичними контекстно-вільними граматиками

В другому розділі було запропоновано нову модель стохастичної контекстно-вільної граматики та доведено еквівалентність моделі та стохастичних контекстно-вільних граматик. Був розброблений програмний пакет на мові C++, шо дозволяє перевірити узгодженість моделі з стохастичними контекстно-вільними граматиками на практиці,

його вихідний код наведений в *додатку А*. Перевіримо емпіричним шляхом узгодженість стохастичних граматик в 2-нормальній формі Грейбах та прихованих моделей Маркова другого порядку за спостереженнями зі стеком.

Приклад 3.1. Нехай задано стохастичну граматику в 2-нормальній формі Грейбах:

$$(0.2)A_1 \to aA_3A_2$$
$$(0.8)A_1 \to cA_2$$
$$(0.5)A_2 \to a$$
$$(0.5)A_2 \to b$$
$$(0.3)A_3 \to dA_2$$
$$(0.3)A_3 \to cA_2$$
$$(0.4)A_3 \to a$$

В результаті для ста речень згенерованих за допомогою стохастичної граматики в 2-нормальній формі Грейбах ми отримали наступний частотний розподіл:

```
a a a : 5
a a b : 1
a c a a : 1
a c a b : 3
a c b a : 2
a c b b : 3
a d b a : 1
a d b b : 1
c a : 45
c b : 38
```

Для речень згенерованих за допомогою побудованої прихованої

моделі Маркова другого порядку зі стеком маємо наступний розподіл:

```
a a a : 1
a a b : 2
a c a a : 1
a c b b : 2
a d a a : 2
a d a b : 2
a d b a : 1
a d b b : 3
c a : 45
c b : 41
```

Як бачимо, статистики по двом згенерованим текстам є дуже близькими позаяк має місце доведена в другому розділі теорема про еквівалентність стохастичних контекстно-вільних граматик та прихованих моделей Маркова другого порядку за спостереженнями зі стеком.

3.3 Перспективи нової моделі

Запропонована модель природної мови, що заснована на прихованій моделі Маркова другого порядку за спостереженнями зі стеком є перспективною з точки зору подальших досліджень в криптоаналізі та обробці природних мов. Її можна застосовувати як модель відкритого тексту, особливо при атаках на криптосистеми, що працюють з текстами природною мовою, наприклад поточні шифри. Її інформативність більша за вже запропоновані розглянуті статистичні моделі мови, позаяк будь-яка природна мова з певною долею точності може бути прийнята за контекстно-вільну. Модель неявно враховує зв'язки між всіма словами у

реченні та семантичну структуру, таким чином статистично суттєво звужуючи можливу множину всіх можливих речень до множини більш імовірних речень. Цим обумовлюється інтерес з точки зору криптоаналізу. Варто зазначити, що модель в плані застосування є надзвичайно ефективною, оскільки складність граматичного виводу лінійно залежить від довжини речення, що виводиться.

Запропонована модель може мати не тільки криптографічне застосування, позаяк в біоінформатиці теж існує потреба в розробці ефективніших алгоритмів навчання ланцюжків ДНК та РНК. Таким чином однією з важливих проблем для даної моделі є розробка нових алгоритмів її навчання за множиною текстів. Для цього можна використати наявний шаблон алгоритму Баума-Велша, модифікувавши його з врахуванням стеку.

Висновки до розділу 3

В розділі розглянуто сучасні підходи до моделювання природної мови та емпіричним шляхом була перевірена узгодженість запропонованої моделі та стохастичних контекстно-вільних граматик. Також було розглянуто перспективи нової моделі та запропоновано ідеї для подальших досліджень.

ВИСНОВКИ

У ході роботи був проведений аналіз наукових джерел за тематикою моделей природних мов та моделей відкритого тексту, формальної теорії мов та граматик, теорії прихованих моделей Маркова,

В роботі вперше запропоновано модель природної мови на основі прихованих моделей Маркова другого порядку зі стеком, разом з цим було доведено теорему про еквівалентність стохастичних регулярних граматик та прихованих моделей Маркова другого порядку за спостереженнями та теорему про еквівалентність стохастичних контекстно-вільних граматик та прихованих моделей Маркова другого порядку за спостереженнями зі стеком. Дані результати є важливими теоретичним фундаментом для розробки нових алгоритмів навчання стохастичних контекстно-вільних граматик.даної

Також запропоновано модель відкритого тексту на основі створеної прихованої моделі Маркова другого порядку за спостереженнями зі стеком, що дозволяє ефективне моделювання граматичної структури природної мови, оскільки модель еквівалентна стохастичним контекстно-вільним граматикам, котрі досить добре можуть описувати граматичну структуру та деякі семантичні властивості мови.

Було розроблено програмний пакет, що дозволяє моделювати стохастичні контекстно-вільні граматики за допомогою прихованих моделей Маркова другого порядку за спостереженнями зі стеком. Зокрема в ньому реалізовано описаний поліноміальний алгоритм зведення стохастичної контекстно-вільної граматики до прихованої моделі Маркова другого порядку за спостереженнями зі стеком.

Емпіричним шляхом було перевірено узгодженість моделі з стохастичними контекстно-вільними граматиками на прикладі довільної граматики.

Розглянута в роботі тематика може бути продовжена в контексті

розробки нових алгоритмів навчання стохастичних контекстно-вільних граматик, оскільки наявні алгоритми мають досить велику обчислювальну складність, а нова модель є простішою у алгоритмічному плані. Також результати роботи можуть бути застосовані в біоінформатиці, як база для розробки нових моделей РНК та ДНК.

ПЕРЕЛІК ПОСИЛАНЬ

1. Hopcroft J. E.; Ullman J. D. Introduction to Automata Theory, Languages, and Computation // Addison-Wesley. - 1979

2. Lari K.; Young S. J. Applications of stochastic context-free grammars using the inside-outside algorithm // Computer Speech and Language. - 1991

3. Lari K.; Young S. J. The estimation of stochastic context-free grammars using the inside-outside algorithm // Computer Speech and Language. - 1990

4. Chomsky, Noam. Three models for the description of language. // IRE Transactions on Information Theory. - 1956

5. Chomsky, Noam. On certain formal properties of grammars // Information and Control - 1959

6. А. М. Яглом; И. М. Яглом. Вероятность и информация // Главное издательство физико-математической литературы, Москва - 1973

7. Greibach, Sheila. A New Normal-Form Theorem for Context-Free Phrase Structure Grammars // Journal of the ACM. 12 - 1965

8. Jurafsky, D.; Martin, J. H. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition // Upper Saddle River, N.J: Pearson Prentice Hall. - 2009

9. Baum, L. E.; Petrie, T.; Soules, G.; Weiss, N. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains // The Annals of Mathematical Statistics. 41 - 1970

10. Baum, L. E.; Petrie, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains // The Annals of Mathematical Statistics. 37 - 1966

11. Sakakibara, Y., Brown, M., Hughley, R., Mian, I., Sjolander, K., Underwood, R., Haussler, D.: Stochastic context-free grammars for tRNA modeling. // Nuclear Acids Res. 22 - 1994

12. From hidden markov models to stochastic contex-free grammars //

Statistics 246, week 9b, spring 2002[Електронний ресурс] Режим доступу — http://www.stat.berkeley.edu/~terry/Classes/s246.2002/Week9/week9b.pdf

13. Viterbi A. J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm // IEEE Transactions on Information Theory. 13 (2) - 1967

14. Eddy S. R.; Durbin R. RNA sequence analysis using covariance models. // Nucleic Acids Research. 22 (11) - 1994

15. Xiaolin Li, Marc Parizeau, Réjean Plamondon: Training Hidden Markov Models with Multiple Observations – A Combinatorial Method // IEEE Transactions on PAMI, vol. PAMI-22, no. 4, pp 371-377 - 2000

16. Robert Koch, Norbert Blum: Greibach Normal Form Transformation, Revisited // Information and Computation Volume 150, Issue 1 - 1999

17. Andrzej Ehrenfeucht, Grzegorz Rozenberg: An Easy Proof of Greibach Normal Form // Information And Control 63, 190-199 - 1984

18. Репозиторій на GitHub[Електронний ресурс] Режим доступу — https://github.com/juja256/hmm2s

19. Uanhua Lv, ChengXiang Zhai: Positional Language Models for Information Retrieval // SIGIR'09 - 2009

20. Ravi Ganesan, Alan T. Sherman: Statistical Techniques for Language Recognition: An Introduction And Guide for Cryptanalysis - 1993

21. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean: Efficient Estimation of Word Representations in Vector Space - 2013 // Режим доступу — https://arxiv.org/pdf/1301.3781.pdf

22. Paul Ibbotson, Michael Tomasello: Evidence Rebuts Chomsky's Theory of Language Learning // Scientific American - 2016

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Тексти інструментальних програм для проведення експериментальних досліджень необхідно виносити у додатки.

А.1 Програмна реалізація прихованих моделей Маркова зі стеком

Лістинг файлу 'hmm.h'

```
#ifndef HMM_H
#define HMM_H

#include <stack>
#include <vector>
#include "tensor.h"
#include "ring_buffer.h"
#include "scfg_gnf.h"

#define NO_OBSERVATION 0xFFFFFFFF
#define HMM_HALT 0xFFFFFFFE
#define EMPTY_STATE 0

unsigned sample_from_categorical(unsigned n, double* pr);

template<int E>
class HMM: public StochasticGrammar {
protected:

  unsigned M, L;
  unsigned cur_observation;
  unsigned takts;
  RingBuffer<unsigned, E> cur_state_buf;

public:
  Tensor<double> transitionPr;
  Tensor<double> observationPr;
  Tensor<double> startPr;
```

```
  HMM(unsigned M, unsigned L);
  HMM(Tensor<double> t, Tensor<double> o, Tensor<double> start);
  ~HMM();
  virtual unsigned start();
  virtual unsigned step();
  virtual unsigned restart();
  virtual unsigned run(unsigned n, unsigned* arr);
  virtual std::vector<unsigned> run(unsigned n);
  virtual std::vector<unsigned> run();
  virtual std::vector<unsigned> infer(unsigned nonterminal=START_SYMBOL);
};

#endif
```

Лістинг файлу 'hmm.cpp'

```
#include "hmm.h"
#include <memory>
#include <cstdlib>
#include <iostream>
#include "settings.h"

unsigned sample_from_categorical(unsigned n, double* pr) {
  std::unique_ptr<double[]> a(new double[n+1]);
  a[0] = 0;
  #ifdef DEBUG
  std::cout << "Sampling␣from␣CDF:␣" << a[0] << "␣";
  #endif
  for (unsigned i = 0; i < n; i++) {
    a[i+1] = a[i] + pr[i];
    #ifdef DEBUG
    std::cout << a[i+1] << "␣";
    #endif
  }
  #ifdef DEBUG
  std::cout << "\n";
  #endif

  double r = a[n] * ((double)rand() / RAND_MAX);
  #ifdef DEBUG
  std::cout << "Random:␣" << r << "\n";
  #endif
  if (r < a[n-1] * 0.5) {
    for (unsigned i=1; i<=n; i++) {
      if (r > a[i-1] && r < a[i]) {
```

```
        return i-1;
      }
    }
  }
  else {
    for (int i=n; i>=1; i--) {
      if (r > a[i-1] && r < a[i])
        return i-1;
    }
  }
}

/* M - state space size */
/* L - observation space size */
/* E - order by emissions */
template<int E>
HMM<E>::HMM(unsigned M_, unsigned L_):
  M(M_), L(L_), takts(0) {
    this->transitionPr = Tensor<double>({this->M, this->M});
    unsigned obs_dims[E+1];
    for (unsigned i=0; i<E; i++)
      obs_dims[i] = M;
    obs_dims[E] = L;
    this->observationPr = Tensor<double>(E+1, obs_dims);
    this->startPr = Tensor<double>({this->M});
}

template<int E>
HMM<E>::HMM(Tensor<double> t, Tensor<double> o, Tensor<double> start):
  M( ((unsigned*)(start.size.bytes))[0] ), L( ((unsigned*)(o.size.bytes))[o.size.getSize()
      -1] ), takts(0) {
  observationPr = o;
  startPr = start;
  transitionPr = t;
}

template<int E>
unsigned HMM<E>::step() {
  this->cur_state_buf.writeTop(
    sample_from_categorical(M, this->transitionPr.getSlice({this->cur_state_buf[0]}))
  );
  this->takts++;

  if (this->takts >= E && this->cur_state_buf[0] != EMPTY_STATE)
    this->cur_observation = sample_from_categorical( this->L, this->observationPr.template
```

```
        getSlice<E>(this->cur_state_buf) );
  else if (this->takts < E)
    this->cur_observation = NO_OBSERVATION;
  else
    this->cur_observation = HMM_HALT;

  #ifdef DEBUG
  std::cout << "Stepping␣HMM" << "(" << this->M << ",␣" << this->L << ").␣Current␣state:␣"
      << this->cur_state_buf[0] << "\n";
  #endif
  return this->cur_observation;
}

template<int E>
unsigned HMM<E>::start() {
  this->cur_state_buf.writeTop(
    sample_from_categorical(M, startPr.getSlice({}) )
  );
  this->takts++;

  if (this->takts >= E && this->cur_state_buf[0] != EMPTY_STATE)
    this->cur_observation = sample_from_categorical(this->L, this->observationPr.template
        getSlice<E>(this->cur_state_buf) );
  else if (this->takts < E)
    this->cur_observation = NO_OBSERVATION;
  else
    this->cur_observation = HMM_HALT;

  #ifdef DEBUG
  std::cout << "Starting␣HMM" << "(" << this->M << ",␣" << this->L << ").␣Start␣state:␣" <<
      this->cur_state_buf[0] << "\n";
  #endif
  return this->cur_observation;
}

template<int E>
unsigned HMM<E>::run(unsigned n, unsigned* arr) {
  unsigned old_takts = this->takts;
  if (this->takts == 0) {
    arr[0] = this->start();
    /* HMM ought not to halt at the start */
  }
  else {
    arr[0] = this->step();
    if (this->cur_state_buf[0] == EMPTY_STATE)
```

```
      goto halt;
  }

  for (unsigned i=1; i<n; i++) {
    arr[i] = this->step();
    if (this->cur_state_buf[0] == EMPTY_STATE)
      goto halt;
  }

halt:
  return this->takts - old_takts;
}

template<int E>
std::vector<unsigned> HMM<E>::run(unsigned n) {
  std::vector<unsigned> v;
  if (this->takts == 0) {
    v.push_back(this->start());
  }
  else {
    v.push_back(this->step());
    if (v.back() == HMM_HALT)
      goto halt;
  }
  for (unsigned i=1; i<n; i++) {
    v.push_back(this->step());
    if (v.back() == HMM_HALT)
      goto halt;
  }

halt:
  return v;
}

template<int E>
std::vector<unsigned> HMM<E>::run() {
  std::vector<unsigned> v;
  if (this->takts == 0) {
    v.push_back(this->start());
  }
  else {
    v.push_back(this->step());
    if (v.back() == HMM_HALT)
      goto halt;
  }
```

```
  for (;;) {
    v.push_back(this->step());
    if (v.back() == HMM_HALT)
      goto halt;
  }

halt:
  return v;
}

template<int E>
unsigned HMM<E>::restart() {
  this->takts = 0;
  return this->start();
}

template<int E>
HMM<E>::~HMM() {

}

template<int E>
std::vector<unsigned> HMM<E>::infer(unsigned nonterminal) {
  std::vector<unsigned> infered = this->run();
  this->takts = 0;
  return infered;
}

template class HMM<1>;
template class HMM<2>;
```

Лістинг файлу 'hmm2s.h'

```
#ifndef HMM2S_H
#define HMM2S_H

#include <stack>
#include "hmm.h"
#include "scfg_gnf.h"

#define HMM2S_HALT HMM_HALT

/* HMM with stack of order 2 by emissions */
class HMM2S: public HMM<2> {
private:
  std::stack<unsigned> stack;
```

```
public:
  HMM2S(unsigned M, unsigned L);
  HMM2S(StochasticGrammarInGNF& grammar);
  virtual unsigned start();
  virtual unsigned step();
  //std::vector<unsigned> infer();
};

#endif
```

Лістинг файлу 'hmm2s.cpp'

```
#include "hmm2s.h"
#include "settings.h"
#include "scfg_gnf.h"
#include <vector>

HMM2S::HMM2S(unsigned M_, unsigned L_) : HMM<2>(M_, L_) {
  this->transitionPr = Tensor<double>({this->M, this->M, this->M});
  this->observationPr = Tensor<double>({this->M, this->M, this->M, this->L});
  this->startPr = Tensor<double>({this->M});
}

/*
  SCFG in GNF to HMM2 with stack converter
  Scenaries:
    A -> A1, Stack.Push A2
    A -> A1
    A -> Stack.Top
*/
HMM2S::HMM2S(StochasticGrammarInGNF& grammar): HMM2S(grammar.getNonterminalCount()+1,
    grammar.getTerminalCount()+1) {
  this->startPr.setElement({START_SYMBOL}, 1.0); // Start state is 1 !
  std::vector<GNF_RULE>& rules = grammar.getRules();
  unsigned size = grammar.getGrammarSize();
  for (unsigned i=0; i<size; i++) {
    /* marginalization: Pr(A -> A1, push A2) = Pr(A -> a A1 A2) + Pr(A -> b A1 A2) */
    this->transitionPr.setElement({rules[i].lhs_nt, rules[i].rhs_nt1, rules[i].rhs_nt2},
      this->transitionPr.getElement({rules[i].lhs_nt, rules[i].rhs_nt1, rules[i].rhs_nt2}) +
           rules[i].probability);
    /* initialization of emission weights from raw rules's probabilities of grammar */
    this->observationPr.setElement({rules[i].lhs_nt, rules[i].rhs_nt1, rules[i].rhs_nt2,
        rules[i].rhs_t}, rules[i].probability);

  }
```

```
}

unsigned HMM2S::start() {
  this->cur_state_buf.writeTop(
    sample_from_categorical(M, startPr.getSlice({}) )
  );
  this->takts++;

  this->cur_observation = NO_OBSERVATION;

  #ifdef DEBUG
  std::cout << "Starting␣HMM2S" << "(" << this->M << ",␣" << this->L << ").␣Start␣state:␣"
  << this->cur_state_buf[0] << "\n";
  #endif
  return this->cur_observation;
}

unsigned HMM2S::step() {
  if ((this->cur_state_buf[0] == EMPTY_STATE) && stack.empty()) {
    #ifdef DEBUG
      std::cout << "Halting␣HMM2S!\n";
    #endif
    return HMM2S_HALT;
  }
  unsigned pos = sample_from_categorical(M*M, this->transitionPr.getSlice({this->
      cur_state_buf[0]}));
  this->cur_state_buf.writeTop(pos / M);
  unsigned for_stack = pos % M;
  if (for_stack != EMPTY_STATE) {
    this->stack.push(for_stack);
  }
  #ifdef DEBUG
    std::cout << "HMM2S␣is␣jumping␣to␣state␣#" << this->cur_state_buf[0] << "␣pushing␣#" <<
        for_stack << "␣to␣stack\n";
  #endif
  this->takts++;

  this->cur_observation = sample_from_categorical(this->L,
    this->observationPr.getSliceForHMM2S(this->cur_state_buf, for_stack)
  );

  if (this->cur_state_buf[0] == EMPTY_STATE && !(this->stack.empty())) {
    this->cur_state_buf[0] = this->stack.top();
    stack.pop();
  }
```

```
  #ifdef DEBUG

  std::cout << "Stepping␣HMM2S" << "(" << this->M << ",␣" << this->L << ").␣Current␣state:␣"
    << this->cur_state_buf[0] << ".␣\n";
  if (!stack.empty())
    std::cout << "Stack␣top:␣" << this->stack.top() << "\n";
  #endif
  return this->cur_observation;
}

/*std::vector<unsigned> HMM2S::infer() {
  return this->run();
}*/
```

Лістинг файлу 'scfl.h'

```
#ifndef SCFL_H
#define SCFL_H

#include <vector>
#include <string>
#include "scfg_gnf.h"
#include <map>

class StochasticLanguage {
  StochasticGrammar* gr;
  std::vector<std::string> vocabulary;
public:
  StochasticLanguage(StochasticGrammar* gr_, std::initializer_list<std::string> v={});
  unsigned addWordToVocabulary(std::string s);
  std::vector<std::vector<std::string>> generateSomeText(unsigned sentanceNum);
  std::vector<std::string> generateSomeSentaces(unsigned sentanceNum, bool withSpaces=true);
  void speakSomeText(unsigned sentanceNum);
  std::map<std::string, unsigned> countFrequencies(unsigned sentanceNum);
};

#endif
```

Лістинг файлу 'scfl.cpp'

```
#include "scfl.h"
#include <iostream>
#include <string>
#include "hmm.h"
#include <map>
```

```
StochasticLanguage::StochasticLanguage(StochasticGrammar* gr_, std::initializer_list<std::
    string> v): gr(gr_){
  vocabulary.push_back("#");
  vocabulary.insert(vocabulary.end(), v);
}
unsigned StochasticLanguage::addWordToVocabulary(std::string s) {
  vocabulary.push_back(s);
  return vocabulary.size();
}
std::vector<std::vector<std::string>> StochasticLanguage::generateSomeText(unsigned
    sentanceNum) {
  std::vector<std::vector<std::string>> text;
  for (unsigned i=0; i<sentanceNum; i++) {
    std::vector<unsigned> raw = gr->infer();
    std::vector<std::string> v;
    for (unsigned j=0; j<raw.size(); j++) {
      if (raw[j] == NO_OBSERVATION || raw[j] == HMM_HALT) continue;
      if (raw[j] > vocabulary.size()) throw StochasticGrammarException(SMALL_VOCABULARY);
      v.push_back(vocabulary[raw[j]]);
    }
    text.push_back(v);
  }
  return text;
}

std::vector<std::string> StochasticLanguage::generateSomeSentaces(unsigned sentanceNum, bool
     withSpaces) {
  std::vector<std::string> text;
  std::string sep = (withSpaces) ? "␣" : "";
  for (unsigned i=0; i<sentanceNum; i++) {
    std::vector<unsigned> raw = gr->infer();
    std::string v;
    for (unsigned j=0; j<raw.size(); j++) {
      if (raw[j] == NO_OBSERVATION || raw[j] == HMM_HALT) continue;
      if (raw[j] > vocabulary.size()) throw StochasticGrammarException(SMALL_VOCABULARY);
      v += (vocabulary[raw[j]]) + sep;
    }
    text.push_back(v);
  }
  return text;
}

void StochasticLanguage::speakSomeText(unsigned sentanceNum) {
  std::vector<std::vector<std::string>> text = this->generateSomeText(sentanceNum);
```

```
  for (unsigned i=0; i<text.size(); i++) {
    for (unsigned j=0; j<text[i].size(); j++) {
      std::cout << text[i][j] << "␣";
    }
    std::cout << ".␣";
  }
  std::cout << "\n";
}

std::map<std::string, unsigned> StochasticLanguage::countFrequencies(unsigned sentanceNum) {
  std::map<std::string, unsigned> freq;
  std::vector<std::string> text = this->generateSomeSentaces(sentanceNum);

  for (unsigned i=0; i<text.size(); i++) {
    if (freq.find(text[i]) == freq.end())
      freq.emplace(text[i], 1);
    else
      freq[text[i]]++;
  }
  return freq;
}
```

Лістинг файлу 'tensor.h'

```
#ifndef TENSOR_H
#define TENSOR_H

#include <initializer_list>
#include <cstdlib>
#include <cstring>
#include "blob.h"
#include <iostream>
#include "ring_buffer.h"

template<class T>
struct Tensor {
  Blob size;
  Blob mem;
  Tensor() {}

  Tensor(std::initializer_list<unsigned> dims) {
    size = Blob(dims.size() * sizeof(unsigned));
    memcpy(size.bytes, dims.begin(), sizeof(unsigned)*dims.size());
    unsigned s=1;
    for (unsigned k: dims)
      s *= k;
```

```
  mem = Blob(s*sizeof(T));
  memset(mem.bytes, 0, s*sizeof(T));
}

Tensor(unsigned dim_size, unsigned* dims) {
  size = Blob(dim_size * sizeof(unsigned));
  memcpy(size.bytes, dims, sizeof(unsigned)*dim_size);
  unsigned s=1;
  for (unsigned k=0; k<dim_size; k++)
    s *= dims[k];
  mem = Blob(s*sizeof(T));
  memset(mem.bytes, 0, s*sizeof(T));
}

unsigned getNumOfDimensions() {
  return size.getSize() / sizeof(unsigned);
}

unsigned getDimension(unsigned idx) {
  return ((unsigned*)(size.bytes))[idx];
}

unsigned getSize() {
  return mem.getSize() / sizeof(T);
}

T getElement(std::initializer_list<unsigned> pos) {
  unsigned index = 0;
  unsigned s = mem.getSize() / sizeof(T);
  for (unsigned i=0; i < pos.size(); i++) {
    //std::cout << ((unsigned*)(size.bytes))[i] << " " << std::flush;
    s /= ((unsigned*)(size.bytes))[i];

    index += s * ((unsigned*)(pos.begin()))[i];
  }
  return ((T*)(mem.bytes))[index];
}

void setElement(std::initializer_list<unsigned> pos, T element) {
  unsigned index = 0;
  unsigned s = mem.getSize() / sizeof(T);
  for (unsigned i=0; i < pos.size(); i++) {
    s /= ((unsigned*)(size.bytes))[i];
    index += s * ((unsigned*)(pos.begin()))[i];
  }
```

```
    ((T*)(mem.bytes))[index] = element;
  }

  T* getSlice(std::initializer_list<unsigned> pos) {
    unsigned index = 0;
    unsigned s = mem.getSize() / sizeof(T);
    for (unsigned i=0; i < pos.size(); i++) {
      s /= ((unsigned*)(size.bytes))[i];

      index += s * ((unsigned*)(pos.begin()))[i];
    }
    return (T*)(mem.bytes) + index;
  }

  template<int N>
  T* getSlice(RingBuffer<unsigned, N>& buf) {
    unsigned index = 0;
    unsigned s = mem.getSize() / sizeof(T);
    for (unsigned i=0; i < N; i++) {
      s /= ((unsigned*)(size.bytes))[i];

      index += s * buf[i+1];
    }
    return (T*)(mem.bytes) + index;
  }

  T* getSliceForHMM2S(RingBuffer<unsigned, 2>& buf, unsigned stackTop) {
    unsigned index = 0;
    unsigned s = mem.getSize() / sizeof(T);
    for (unsigned i=0; i < 2; i++) {
      s /= ((unsigned*)(size.bytes))[i];
      index += s * buf[i+1];
    }

    s /= ((unsigned*)(size.bytes))[2];
    index += s * stackTop;

    return (T*)(mem.bytes) + index;
  }

};

#endif
```

Лістинг файлу 'scfg_gnf.h'

```
#ifndef SCFG_GNF_H
#define SCFG_GNF_H

#define EMPTY_SYMBOL 0
#define START_SYMBOL 1

/* Exceptions */
#define PROBABILITY_READ_ERR 1
#define LHS_NT_READ_ERR 2
#define SEPARATOR_READ_ERR 3
#define SEPARATOR_INVALID 4
#define RHS_T_READ_ERR 5
#define GNF_READ_ERR 6
#define NT_INVALID 7
#define T_INVALID 8
#define SMALL_VOCABULARY 9

#include <vector>

typedef struct {
  double probability;
  unsigned lhs_nt;
  unsigned rhs_t;
  unsigned rhs_nt1;
  unsigned rhs_nt2;
} GNF_RULE; // 24 bytes

unsigned sample_from_categorical(unsigned n, GNF_RULE* pr);

class StochasticGrammarException {
  unsigned e;
public:
  StochasticGrammarException(unsigned err) : e(err) {}
  int getErrorCode() { return e; }
};

class StochasticGrammar {
public:
  virtual std::vector<unsigned> infer(unsigned nonterminal=START_SYMBOL)=0;
  virtual ~StochasticGrammar() {}
};

class StochasticGrammarInGNF : public StochasticGrammar {
  unsigned M, L;
  std::vector<GNF_RULE> rules;
```

```
  std::vector<unsigned> rules_cnt;
public:
  StochasticGrammarInGNF(const char* fn);
  ~StochasticGrammarInGNF();
  std::vector<unsigned> infer(unsigned nonterminal=START_SYMBOL);
  unsigned getGrammarSize();
  unsigned getTerminalCount();
  unsigned getNonterminalCount();
  std::vector<GNF_RULE>& getRules();
};

#endif
```

Лістинг файлу 'scfg_gnf.cpp'

```
#include "scfg_gnf.h"
#include <sstream>
#include <string>
#include <stack>
#include <algorithm>
#include <vector>
#include <memory>
#include <fstream>
#include <iostream>
#include <ctime>
#include "settings.h"

unsigned sample_from_categorical(unsigned n, GNF_RULE* rules) {
  std::unique_ptr<double[]> a(new double[n+1]);
  a[0] = 0;
  #ifdef DEBUG
  std::cout << "Sampling␣from␣CDF:␣" << a[0] << "␣";
  #endif
  for (unsigned i = 0; i < n; i++) {
    a[i+1] = a[i] + rules[i].probability;
    #ifdef DEBUG
    std::cout << a[i+1] << "␣";
    #endif
  }
  #ifdef DEBUG
  std::cout << "\n";
  #endif

  double r = a[n] * ((double)rand() / RAND_MAX);
  #ifdef DEBUG
  std::cout << "Random:␣" << r << "\n";
```

```
  #endif
  if (r < a[n-1] * 0.5) {
    for (unsigned i=1; i<=n; i++) {
      if (r > a[i-1] && r < a[i]) {
        return i-1;
      }
    }
  }
  else {
    for (int i=n; i>=1; i--) {
      if (r > a[i-1] && r < a[i])
        return i-1;
    }
  }
}

StochasticGrammarInGNF::StochasticGrammarInGNF(const char* fn) {
  srand(time(NULL));
  std::ifstream file(fn);
  std::string line;
  unsigned n=2;
  std::getline(file, line);

  std::stringstream linestream(line);
  std::string gnf;
  if (!(linestream >> gnf) || gnf != "GNF") {
    throw StochasticGrammarException( GNF_READ_ERR | (1 << 8) );
  }
  unsigned m,l;
  if (!(linestream >> m >> l)) {
    throw StochasticGrammarException( GNF_READ_ERR | (1 << 8) );
  }
  else {
    this->M = m;
    this->L = l;
  }


  while(std::getline(file, line) && line[0] != '#') {
    std::stringstream linestream(line);
    std::string separator;
    GNF_RULE cur_rule;

    if (!(linestream >> cur_rule.probability)) {
      throw StochasticGrammarException(PROBABILITY_READ_ERR | (n >> 8));
```

```
  };
  if (!(linestream >> cur_rule.lhs_nt)) {
    throw StochasticGrammarException(LHS_NT_READ_ERR | (n >> 8));
  }
  if (!(linestream >> separator)) {
    throw StochasticGrammarException(SEPARATOR_READ_ERR | (n >> 8));
  }
  if (separator != "->") {
    throw StochasticGrammarException(SEPARATOR_INVALID | (n >> 8));
  }
  if (!(linestream >> cur_rule.rhs_t)) {
    throw StochasticGrammarException(RHS_T_READ_ERR | (n >> 8));
  }

  /* tail of the rule */
  if (!(linestream >> cur_rule.rhs_nt1)) {
    cur_rule.rhs_nt1 = EMPTY_SYMBOL;
  }
  if (!(linestream >> cur_rule.rhs_nt2)) {
    cur_rule.rhs_nt2 = EMPTY_SYMBOL;
  }

  if ( (cur_rule.lhs_nt > this->M) || (cur_rule.rhs_nt1 > this->M) || (cur_rule.rhs_nt2 >
      this->M) ) {
    throw StochasticGrammarException(NT_INVALID | (n >> 8));
  }
  if ( cur_rule.rhs_t > this-> L ) { /* Important: Terminal Alphabet starts from 1 !!! */
    throw StochasticGrammarException(T_INVALID | (n >> 8));
  }
  n++;
  rules.push_back(cur_rule);
}
/* sorting rules with respect to lhs nonterminals */
std::sort(rules.begin(), rules.end(), [](const GNF_RULE & a, const GNF_RULE & b) -> bool {
     return a.lhs_nt < b.lhs_nt; });
this->rules_cnt.push_back(0);

/* iterators to each lhs nonterminal */
for (unsigned i=1; i<rules.size(); i++) {
  if (rules[i-1].lhs_nt != rules[i].lhs_nt) {
    rules_cnt.push_back(i);
  }
}

this->rules_cnt.push_back(rules.size());
```

```
  #ifdef DEBUG
  std::cout << "Rules␣indices:␣";
  for(unsigned i=0;i<rules_cnt.size();i++) {
    std::cout << rules_cnt[i] << "␣";
  }
  std::cout << "\n";
  #endif
}

StochasticGrammarInGNF::~StochasticGrammarInGNF() { }

std::vector<unsigned> StochasticGrammarInGNF::infer(unsigned nonterminal) {
  std::stack<unsigned> state;
  std::vector<unsigned> sentance;
  state.push(nonterminal);
  while(!state.empty()) {
    unsigned cur_state = state.top();
    #ifdef DEBUG
      std::cout << "State␣before␣inference␣#" << cur_state << "\n";
    #endif
    state.pop();
    unsigned sampled = sample_from_categorical(this->rules_cnt[cur_state] - this->rules_cnt[
        cur_state-1],
    rules.data() + this->rules_cnt[cur_state-1]);
    #ifdef DEBUG
      std::cout << "Sampled␣raw:␣" << sampled << "\n";
    #endif
    unsigned cur_rule = this->rules_cnt[cur_state-1] + sampled;

    #ifdef DEBUG
      std::cout << "Rule␣#" << cur_rule << "␣is␣selected␣for␣inference\n" << std::flush;
    #endif
    if (this->rules[cur_rule].rhs_nt2) {
      state.push(this->rules[cur_rule].rhs_nt2);
    }
    if (this->rules[cur_rule].rhs_nt1) {
      state.push(this->rules[cur_rule].rhs_nt1);
    }
    sentance.push_back(this->rules[cur_rule].rhs_t);
  }
  return sentance;
}

unsigned StochasticGrammarInGNF::getGrammarSize() {
```

```
  return rules.size();
}

unsigned StochasticGrammarInGNF::getTerminalCount() {
  return L;
}

unsigned StochasticGrammarInGNF::getNonterminalCount() {
  return M;
}

std::vector<GNF_RULE>& StochasticGrammarInGNF::getRules () {
  return rules;
}
```

Лістинг файлу 'ring_buffer.h'

```
#ifndef RING_BUFFER_H
#define RING_BUFFER_H
#include <iostream>
static unsigned true_mod(int a, unsigned N) {
  if (a>=0) return a % N;
  else return a%N + N;
}

template<class T, int N>
struct RingBuffer {
  T buf[N];
  unsigned top;

  RingBuffer(): top(0) {}
  void writeTop(T e) {
    top = true_mod(top+1, N); // top + 1 % N;
    buf[top] = e;
  }
  T& operator[](int i) {
    return buf[true_mod(top + i , N)];
  }
};

#endif
```

Лістинг файлу 'blob.h'

```
#ifndef BLOB_H
#define BLOB_H
```

```
class Blob {
public:
  union {
    char* data;
    unsigned char* bytes;
  };

  Blob();
  Blob(const char* text);
  Blob(const unsigned char* t, int size);
  Blob(int size);
  Blob(const Blob& b2);
  unsigned getSize();
  void operator=(const Blob& b2);
  void clear();
  Blob copy();
  ~Blob();
};

bool saveBlob(Blob& b, const char* fileName);
Blob loadBlob(const char *fileName);

#endif
```

Лістинг файлу 'blob.cpp'

```
#include "blob.h"
//#include "long_ar.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

typedef struct {
  unsigned ref_cnt;
  unsigned size;
} BLOB_HEADER;

static struct {
  BLOB_HEADER hdr;
  unsigned char data;
} emptyBlob = {{ 1,0 }, 0};

static int inc_ref_cnt(unsigned char* d) {
  ((BLOB_HEADER*)d - 1)->ref_cnt++;
  return ((BLOB_HEADER*)d - 1)->ref_cnt;
}
```

```
static int dec_ref_cnt(unsigned char* d) {
  ((BLOB_HEADER*)d - 1)->ref_cnt--;
  return ((BLOB_HEADER*)d - 1)->ref_cnt;
}

bool saveBlob(Blob& data, const char *fileName) {
  FILE *f = fopen(fileName, "wb");
  if (!f)  return false;
  bool success = (fwrite(data.bytes, data.getSize(), 1, f) != 0);
  fclose(f);
  return success;
}

Blob loadBlob(const char *fileName) {
  FILE *f = fopen(fileName, "rb");
  if (!f)  return Blob();
  fseek(f, 0, SEEK_END);
  long fileSize = ftell(f);
  fseek(f, 0, SEEK_SET);
  Blob res(fileSize);
  if (!fread(res.bytes, fileSize, 1, f))
    res = Blob();
  fclose(f);
  return res;
}

Blob::Blob() {
  this->bytes = &(emptyBlob.data);
  inc_ref_cnt(this->bytes);
}

Blob::Blob(const char* text) {
  unsigned l = strlen(text);
  BLOB_HEADER* h = (BLOB_HEADER*)(new unsigned char[l + 1 + sizeof(BLOB_HEADER)]);
  h->size = l;
  h->ref_cnt = 1;
  this->bytes = (unsigned char*)(h + 1);
  strcpy(this->data, text);
  this->bytes[l] = 0;
}

Blob::Blob(const unsigned char* t, int size) {
  BLOB_HEADER* h = (BLOB_HEADER*)(new unsigned char[size + 1 + sizeof(BLOB_HEADER)]);
  h->size = size;
```

```
  h->ref_cnt = 1;
  this->bytes = (unsigned char*)(h + 1);
  memcpy(this->data, t, size);
  this->bytes[size] = 0;
}

Blob::Blob(int size) {
  BLOB_HEADER* h = (BLOB_HEADER*)(new unsigned char[size + 1 + sizeof(BLOB_HEADER)]);
  h->size = size;
  h->ref_cnt = 1;
  this->bytes = (unsigned char*)(h + 1);
  this->bytes[size] = 0;
}

Blob::Blob(const Blob& b2) {
  inc_ref_cnt(b2.bytes);
  this->data = b2.data;
}

void Blob::operator=(const Blob& b2) {
  inc_ref_cnt(b2.bytes);

  if (dec_ref_cnt(this->bytes) == 0) {
    delete[] (unsigned char*)((BLOB_HEADER*)(this->bytes) - 1);
  }

  this->data = b2.data;
}

void Blob::clear() {
  for (int i = 0; i < this->getSize(); i++) {
    this->bytes[i] = 0;
  }
}

Blob Blob::copy() {
  if (this->getSize())
  return Blob(this->bytes, this->getSize());
  else return Blob();
}

Blob::~Blob() {
  if (dec_ref_cnt(this->bytes) == 0) {
    delete[] (unsigned char*)((BLOB_HEADER*)(this->bytes) - 1);
  }
```

```
}

unsigned Blob::getSize() {
  return ((BLOB_HEADER*)(this->bytes) - 1)->size;
}
```

Лістинг файлу 'test.cpp'

```
#include "tensor.h"
#include "ring_buffer.h"
#include <iostream>
#include "hmm.h"
#include "hmm2s.h"
#include "scfg_gnf.h"
#include "scfl.h"
#include <map>

std::ostream& printTensorSlice(std::ostream& out, Tensor<double>& t, unsigned pos, unsigned
    level) {
  unsigned cur_dim_idx = t.getNumOfDimensions() - level - 1;
  unsigned block_sz = t.getSize();

  for (unsigned i = 0; i<=cur_dim_idx; i++) {
    block_sz /= t.getDimension(i);
  }

  if (level == 0) {
    for (unsigned i=0; i < t.getDimension(cur_dim_idx); i++) {
      out << ((double*)(t.mem.bytes))[pos+i] << "␣";
    }
    return out;
  }
  else {
    //out << "level# " << level << ", pos# " << pos << "\n";
    //out << "cur_dim_idx " << cur_dim_idx << ", block_sz " << block_sz << "\n";
    for (unsigned i=0; i < t.getDimension(cur_dim_idx); i++) {
      printTensorSlice(out, t, pos + i*block_sz, level-1);
      out << "\n";
    }
  }
}

std::ostream& operator<<(std::ostream& out, Tensor<double>& t) {
  out << "Tensor<double>,␣dims={";
  for (unsigned i=0; i<t.getNumOfDimensions(); i++)
    out << t.getDimension(i) << ",␣";
```

```
  out << "}\n";
  printTensorSlice(out, t, 0, t.getNumOfDimensions()-1);
  out << "\n";
  return out;
}

void test1() {
  RingBuffer<unsigned, 2> buf;
  buf.writeTop(5);
  std::cout << buf[0] << "␣" << buf[1] << "\n";
  buf.writeTop(4);
  std::cout << buf[0] << "␣" << buf[1];

}

void test2() {
  StochasticGrammarInGNF gr("ex2.gnf");
  std::cout << "###␣HMM␣with␣stack␣test␣###\n";
  HMM2S hmm(gr);

  std::cout << hmm.startPr << hmm.transitionPr << hmm.observationPr;
  std::vector<unsigned> s = hmm.run();
  std::cout << "Infered␣sentance␣of␣length␣" << s.size() << "\n";
  for (unsigned i=0; i<s.size(); i++) {
    std::cout << s[i] << "␣";
  }
  std::cout << "\n";
}

void test3() {
  try{
    StochasticGrammarInGNF gr("ex2.gnf");
    std::cout << "###␣SCFG␣in␣GNF␣test␣###\n" <<
                 "size␣=␣" << gr.getGrammarSize() << "\n" <<
                 "M␣=␣" << gr.getNonterminalCount() << "\n" <<
                 "L␣=␣" << gr.getTerminalCount() << "\n";
    std::vector<unsigned> s = gr.infer();
    std::cout << "Infered␣sentance␣of␣length␣" << s.size() << "\n";
    for (unsigned i=0; i<s.size(); i++) {
      std::cout << s[i] << "␣";
    }
    std::cout << "\n";
  }
  catch(StochasticGrammarException& e) {
    std::cout << "Error␣occured␣#" << std::hex << e.getErrorCode() << "\n";
```

```
  }

}

void test4() {
  StochasticGrammarInGNF first_grammar_provider("ex2.gnf");
  HMM2S second_grammar_provider(first_grammar_provider);
  std::initializer_list<std::string> vocabulary = {"a", "b", "c", "d"};
  StochasticLanguage lang1(&first_grammar_provider, vocabulary);
  StochasticLanguage lang2(&second_grammar_provider, vocabulary);
  std::cout << "SCFG␣in␣GNF␣provider␣generated␣language:␣\n";
  lang1.speakSomeText(100);

  std::cout << "HMM2S␣provider␣generated␣language:␣\n";
  lang2.speakSomeText(100);
}

void test5() {
  StochasticGrammarInGNF first_grammar_provider("ex2.gnf");
  HMM2S second_grammar_provider(first_grammar_provider);
  std::initializer_list<std::string> vocabulary = {"a", "b", "c", "d"};
  StochasticLanguage lang1(&first_grammar_provider, vocabulary);
  StochasticLanguage lang2(&second_grammar_provider, vocabulary);
  std::cout << "SCFG␣in␣GNF␣provider␣generated␣language:␣\n";
  std::map<std::string, unsigned> freq1 = lang1.countFrequencies(100);
  for (auto& x: freq1) {
    std::cout << x.first << ":␣" << x.second << "\n";
  }

  std::cout << "HMM2S␣provider␣generated␣language:␣\n";
  std::map<std::string, unsigned> freq2 = lang2.countFrequencies(100);
  for (auto& x: freq2) {
    std::cout << x.first << ":␣" << x.second << "\n";
  }
}

int main() {
  //test4();
  test5();
  return 0;
}
```

Лістинг файлу 'ex2.gnf'

```
GNF 3 4
0.2 1 -> 1 3 2
```

```
0.8 1 -> 3 2
0.5 2 -> 1
0.5 2 -> 2
0.3 3 -> 4 2
0.3 3 -> 3 2
0.4 3 -> 1
```