Короткий вступ до питання побудови текстів з заданими фонетичними властивостями (з використанням Haskell програм)

Олександр Сергійович Жабенко

22 лютого 2022 р.

Автор та розробник ПЗ: Олександр Сергійович Жабенко

Анотація

У роботі запропоновано оригінальний підхід до завдання побудови текстів з заданими фонетичними властивостями, який дозволяє ефективно та швидко працювати над задачею, пропонуючи цікаві рішення як у змістовній, так і в інженерній площинах. Побудовано елементи теорії, яка проілюстрована використанням програм мовою Haskell для значного прискорення необхідних обчислень.

Вступ

Існують різні мови. У них є різна структура та правила. При цьому існує можливість створити та використовувати (на основі однієї з існуючих широко вживаних та добре розповсюджених мов, зокрема української у цій роботі) “фонетичну” мову, яка краще підходить для поезії та музики. Можливо навіть створити різні варіанти фонетичної мови. Ця робота пропонує створити кілька різних фонетичних мов на основі української.

Уявіть, що ви можете розуміти інформацію в тексті незалежно від порядку слів і при збереженні лише найбільш необхідної граматики (наприклад, правило не відокремлювати прийменник та наступне слово збережене). Розуміти точно так само, як читати текст (після деякого навчання та тренування, можливо), в якому у словах збережені на своїх позиціях лише перші та останні літери, а всі решта – взаємно перемішані одна з одною. Отже, уявіть, що ви можете розуміти (і виражати ваші думки, почуття, мотиви тощо) повідомлення тексту без дотримання строгого порядку слів.

У такому випадку ви можете впорядкувати слова (зберігши найбільш необхідну граматику для зменшення чи повного усунення можливої двозначності, зумовленої граматикою, точніше зменшенням її обсягу), розмістивши їх таким чином, щоб вони забезпечили собою більш цікаве фонетичне звучання. Ви можете спробувати створити поетичний (чи принаймні дещо більш ритмічний та виразний) текст чи музику. Це також може бути саме по собі розвивальною вправою, яка надихає. Але як би ви могли швидко знайти, які комбінації більш чи менш підходять? Крім того, чи може складність алгоритмів бути зменшена?

Це лише деякі з цікавих питань. Ця праця на даний момент не дає повної відповіді на них, але є експериментальною та дослідницькою, при цьому, звичайно, будь-який результат її є цінним.

Українська є мовою без строгих вимог до порядку слів у реченні (хоча є певні усталені переважні варіанти) і має приємне звучання. Отже, вона може бути гарним прикладом та зразком. Крім того, для автора програм це рідна мова.

Навіть якщо ви не бажаєте створити та використовувати “фонетичні” мови, де фонетика є більш важливою, ніж граматика, і тоді ви можете оцінити фонетичний потенціал слів, використаних у тексті, для продукування спеціальним чином озвучених

текстів. Це також може бути цінним та помічним у написанні поезії та можливих інших пов’язаних областях.[68]

Тривалості звуків як основа ритмічності

Основою підходу є той факт, що звуки мови мають різну тривалість, яка залежить від багатьох факторів, яка визначається зокрема як способом звукоутворення (різним для кожного з них), так і іншими факторами, які тією чи іншою мірою можна контролювати, але, зазвичай, повний контроль не вимагається і не досягається. Це призводить до того, що чередування звуків, зокрема їх поєднання у склади, утворює певний ритмічний малюнок. Людина має здатність (яка піддається розвитку та тренуванню) розпізнавати риси цього малюнку, порівнювати їх між собою, робити певні фонетично-ритмічні узагальнення.

Питання визначення тривалостей звуків мови є непростим, але точний результат як уже зазначалося не вимагається. У даній реалізації підходу фонетичних мов використано певні статистичні характеристики звуків, зокрема визначено можливі тривалості. Якщо порівняти спосіб визначення тривалостей, який запропоновано та застоовано у програмі пакету r-glpk-phonetic-languages-ukrainian-durations, то аналогією буде упаковка об’ємних об’єктів. Для спостерігача упаквока буде уявною моделлю процесу отримання тривалостей звуків. Програма pldUkr (її узагальнення pldPL з пакету phonetic-languages-phonetics-basics також слідує цим шляхом, але у ній немає нормування, оскільки для різних мов може не бути такого явища як палаталізація) методом лінійного програмування шукає мінімальну оболонку (не в строгому математичному значенні), всередині якої можуть «вміститися» звуки мови. Ця оболонка і має аналогією упаковку, у той час як звуки мови мають аналогією об’єкти змінного об’єму всередині упаковки. Один і той же звук може вживатися у різних ситуаціях, у різних словах з різною тривалістю, але програма старається підібрати такі тривалості, які б «охоплювали» (подібно як огинаюча крива «охоплює» те чи інше сімейство кривих) усі ці варіації для усіх звуків, при цьому застосовується додатково певне нормування на тривалість фонетичного явища палаталізації (пом’якшення) приголосного, яке менше всього контролюється людиною, а тому очікується, що ця тривалість є найбільш стійкою до можливих випадкових чи систематичних коливань. Для української мови можлива тривалість, яка не видозмінює сильно звучання визначена експериментально з використанням комп’ютерної програми mm1.

Зрештою, нормування не є обов’язковим, важливо, щоб усі тривалості були пропорційні одна одній, тобто важливими є не самі тривалості (що чисельно виражаються як дійсні додатні числа), а їх взаємні співвідношення (допускається множення цих тривалостей одночасно всіх на одне і те ж додатнє число, що не впливає на результати підходу).

Поліритм як мультивпорядкована послідовність

Нехай маємо послідовність з наступною структурою. Нехай ми здійснили (взагалі кажучи умовний) поділ послідовності на компактні однозв'язні підгрупи з однаковою кількістю елементів кожна у підгрупі, що фактично означає, що ми розбили послідовність на послідовність підпослідовностей з однаковою кількістю елементів у кожній. Розглянемо внутрішню упорядкованість кожної підпослідовності у розумінні розміщення значень її елементів (їх можна порівнювати за відношенням порядку, тобто вони є типом даних, який має реалізований екземпляр класу Ord) та повторюваності елементів. Вважаючи, що елементи підпослідовностей можуть бути попарно різними (чи в окремих випадках і однаковими), будемо порівнювати позиції, на яких розміщуються в підпослідовностях підгрупи елементів, які між собою мають вищий ступінь спорідненості (”близькості”, ”схожості”, ”подібності”) за величиною та порядком (якщо це тип даних числовий, який має природний порядок, наприклад, Double, тоді поняття ”спорідненості” за величиною та ”спорідненості” за порядком співпадають). Позначимо такі підгрупи індексами, які в коді модулів мають частіше всього буквенне позначення.

Тоді кожна підпослідовність складатиметься з однакової кількості елементів однієї природи (зокрема, чисел типу Double), у кожній підпослідовності будуть виділені кілька підгруп ”подібних” елементів за величиною (і порядком, якщо підпослідовності відсортувати за величиною), кожна з яких матиме свій індекс у вигляді символу (найчастіше у коді – літери). Підгрупи мають мати (насправді приблизно) однакову кількість елементів (у коді це не витримано строго для спрощення останнього, але це так у переважній більшості випадків в силу надлишкової ”точності” чисел типу Double, що використовуються). Розглянемо питання, на яких позиціях у підпослідовностях розміщуються елементи з тих же самих підгруп, але які належать до різних підпослідовностей.

Для оцінки цього введемо певні числові функції, які мають регулярну поведінку та дозволяють визначати за їх результатом, чи підпослідовності мають елементи, які належать до відповідних підгруп на тих же самих місцях, чи на різних. Можна показати, що ситуація ”на різних” відповідає наявності кількох ритмів – для кожної підгрупи буде свій власний, які між собою не співпадають, водночас ідеальна ситуація ”повністю на тих самих місцях” відповідає випадку, коли ці ритми між собою узгоджуються, подібно як це відбувається у випадку когерентності у квантовій фізиці, зокрема просторової та часової когерентності, що є важливим зокрема для розуміння роботи лазерів та мазерів. Поліритми, які між собою когерують, утворюють помітніший загальний ритм, подібно як і наявність когерентності у випромінюванні веде до появи більшої структурованості випромінювання.

Як ілюстрація до ідей розділу наступні дані.

Приклад ритмічної послідовності (ідеальний випадок).

```
Prelude Rhythmicity.PolyRhythm Numeric> let f x = putStrLn . showFFloat (Just 4) (sin (2*pi*x)) $ ""
   in mapM_ f [0,0.2..4]
0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000
0.9511
0.5878
-0.5878
-0.9511
-0.0000

Prelude Rhythmicity.PolyRhythm Numeric> getPolyChRhData 'a' 5 (PolyCh [True,True,True,False] 5)
  (PolyRhythm [1,1,1,1,1]) . map (sin . (*pi) . (*2)) $ [0,0.2..4]
[[RP P c,RP P a,RP P b,RP P e,RP P d],[RP P c,RP P a,RP P b,RP P e,RP P d],[RP P c,RP P a,
  RP P b,RP P e,RP P d],[RP P c,RP P a,RP P b,RP P e,RP P d]]
```

А ось приклад (мало-)неритмічної послідовності.

```
Prelude Rhythmicity.PolyRhythm Numeric> let f x = putStrLn . showFFloat (Just 4)
  (sin (27182.81828459045*pi*x)) $ "" in mapM_ f [0,0.01..0.24]
0.0000
-0.5139
-0.8817
-0.9988
-0.8319
-0.4284
0.0969
0.5947
0.9233
0.9894
0.7742
0.3388
-0.1930
-0.6698
-0.9562
-0.9707
-0.7092
-0.2460
0.2872
0.7386
0.9801
0.9428
0.6375
0.1509
-0.3787
Prelude Rhythmicity.PolyRhythm Numeric> getPolyChRhData 'a' 5 (PolyCh [True,True,True,False] 5)
  (PolyRhythm [1,1,1,1,1]) . map (sin . (*27182.81828459045) . (*pi)) $  [0,0.01..0.24]
```

```
[[RP P a,RP P b,RP P e,RP P d,RP P c],[RP P d,RP P e,RP P c,RP P b,RP P a],[RP P a,RP P b,
  RP P c,RP P e,RP P d],[RP P d,RP P e,RP P c,RP P b,RP P a],[RP P a,RP P b,RP P c,RP P e,RP P d]]
```

Когерентні стани поліритмічності як одне з суттєвих джерел ритмічності

Описаний паттерн виникнення ритмічності є одним з суттєвих можливих варіантів утворення ритмічності зокрема у текстах чи музиці, але не єдиним. Варто зазначити, що описаний механізм утворення ритмічності, як показують статистичні експерименти з текстами з використанням цього коду (коду бібліотеки та залежних від неї пакетів на сайті Hackage), може бути не єдиним можливим варіантом, але у багатьох випадках має визначальне значення та вплив на перебіг процесу ритмізації (утворення, зміни чи зникнення ритму). Також відомо, що наявність статистичного взаємозв'язку не означає наявності більш глибоких видів зв'язку між явищами, зокрема причинно-наслідкових. "Кореляція не означає причинності". Більш глибокий зв'язок передбачає наявність інших, не статистичних даних, які дозволяють його підтвердити.

Наслідки для реп музики

Код бібліотеки дозволяє на практиці отримати ритмічні зразки, які близькі часто до текстів пісень у стилі реп. Тому це можна віднести до одного з прямих застосувань бібліотеки.[70]

Дитина або хтось новий для мови вчиться читати

Коли дитина тільки починає читати слова мови (або це може бути хтось новий щодо мови). він чи вона починають з вимови фонем для кожного значимого написаного (і, отже, прочитаного) символу. Згодом після деякої практики, дитина починає читати рівно, більш плавно. Тим не менше, якщо текст власне є поетичним твором, зокрема віршованим, ЧАСТО (можливо, зазвичай, чи іноді, чи частіше тощо) є очевидним, що текст, який ось читається у такій манері, має деякі ритмічні властивості, незважаючи на те що фонеми читаються і вимовляються у режимі нерегулярних і до певної міри невідповідних для нормального мовлення тривалостей. Нам буває (часто) достатньо певної організації елементів тексту, щоб відрізнити поетичий текст від непоетичного.

Подібна ж ситуація виникає, коли особа з акцентом (можливо, сильним чи доволі незвичним) читає поетичний текст. Або також в інших ситуаціях. Бібліотека спроектована таким чином, наче Ви маєте справу саме з такими ситуаціями. У ній вважається, що тривалості є фіксованими і відомими (заданими) наперед, і тоді наявність кількох обґрунтованих (більш чи менш) дозволяє оцінити (звичайно, наближено) ритмічні властивості та деякі інші, відповідно до запропонованих тут алгоритмів.

Це все, на думку автора, є підґрунтям для використання бібліотеки та її функціональності в таких випадках.[69]

Функції для збільшення та зменшення

Починаючи з версії пакету phonetic-languages-rhythmicity 0.5.3.0, функції для збільшення та зменшення для поліритмічності стали більш подібними до взаємно обернених функцій. Це, очікувано, веде до більш "гладкої" поведінки на початку рядка, в середині та в кінці.

Зауваження: Починаючи з версії 0.6.0.0 пакету phonetic-languages-rhythmicity значення властивостей серій "c", "s", "t", "u", "v", а також багатьох інших (починаючи з версії 0.9.0.0) можуть бути від'ємними, це не впливає на загальну логіку роботи програм. Починаючи з версії 0.8.0.0 пакету phonetic-languages-simplified-examples-array, додано також нові властивості, які також можуть бути від'ємними за знаком.

Питання вибору найкращої функції та суміжні питання

Розглянемо наступне питання: припустимо, ми отримали найкращий варіант (на нашу суб'єктивну думку чи на основі якихось критеріїв, це тут несуттєво) рядка тим чи іншим способом (тут спосіб насправді ролі не грає). Чи існує така функція, яка саме цей варіант рядка робить оптимальним, тобто для якої саме такий варіант рядка з поміж усіх можливих перестановок дає максимум? Так, існує. Це нескладно довести. Доведення нагадує принцип роботи еквалайзерів.

Нехай $n \in N$ – це кількість складів у такому рядку. Розташуємо тривалості складів у ряд за зростанням (стандартна процедура для описової статистики). Знайдемо найменшу ненульову різницю між сусідніми значеннями, поділимо її на 5. Позначимо цю отриману величину δ. Тепер розглянемо ряд тривалостей складів саме для нашого найкращого рядка. Пронумеруємо кожний склад від початку, рахуючи від 1. Позначимо $Y = \{y_i, \quad i \in N, \quad i = 1, 2, \ldots, n\}$ множину всіх значень тривалостей в порядку слідування складів у найкращому рядку. Позначимо $X = \{0 = x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_{n+1}, \quad i \in N, \quad i = 1, 2, \ldots, n\}$ множину коор-

динат точок кінців часових умовних інтервалів, на які поділяє наш найкращий рядок часову пряму (лівий край рівний 0, бо відлік часу починається з 0). Позначимо $M = \{z_i = \frac{x_i + x_{i+1}}{2}, \quad i \in N, \quad i = 1, 2, \ldots, n-1\}$ множину середин відрізків, на які поділяють часову пряму кінці умовних інтервалів. Позначимо $L_1[a, b]$ клас функцій, інтегровних за Лебегом на відрізку $[a, b]$. Позначимо $I(y_i, z_i, \delta)[z_i - \delta, z_i + \delta]$ клас усіх обмежених функцій з $L_1[z_i - \delta, z_i + \delta]$, максимальне та мінімальне значення кожної з яких лежить на відрізку $[y_i - \delta, y_i + \delta]$. Кожну функцію класу I позначимо g.

Розглянемо клас функцій F (своєрідних скінченних аналогів відомих дельта-функцій Дірака), визначених наступним чином:

$$f(x,i) = \begin{cases} g \in I(y_i, z_i, \delta)[z_i - \delta, z_i + \delta], & \text{якщо } y_i \text{ є унікальним значенням у множині } Y,\ x \in [z_i - \delta, z_i + \delta] \\ y_i, & \text{якщо } y_i \text{ має рівне значення з якимось іншим числом з множини } Y,\ x \in [z_i - \delta, z_i + \delta] \\ 0, & \text{в усіх інших точках} \end{cases}$$

Легко переконатися, що

$$\sum_{i=1}^{n} \int_{-\infty}^{\infty} f(x,i)\, \mathrm{d}x,$$

де інтегрування здійснюється за Лебегом, і є шуканою функцією (бо лише склади найкращого рядка на своєму місці, враховані зі своїми індексами, дають додатній внесок у її значення, а для всіх інших варіантів функції хоча б деякі зі складів дають 0 внесок), причому вона не єдина в силу того, що у першому рядку визначення $f(x,i)$ функція може мати довільне значення з замкненого непорожнього інтервалу. Отже, існує принаймні один клас функцій, який описується такою формулою, для кожної з яких даний варіант рядка буде оптимальним.

Поставимо наступне питання: якщо розглянути не рядок, а їх сукупність, наприклад, вірш чи поему. Чи буде існувати для усього твору (для усієї цієї сукупності рядків) функція, яка буде робити оптимальним кожний рядок, тобто яка буде описувати увесь твір, кожний рядок у ньому?

У даному випадку, попередній спосіб побудови функції не дає бажаного результату, оскільки вже для двох рядків може бути, що те, що є кращим для одного з них, є не найкращим для іншого. У випадку збільшення кількості рядків ця загальна неоптимальність тільки посилюється. Тим не менше, існування такої функції для різних частинних випадків є принципово можливою ситуацією, щоправда, імовірність її зменшується як зі збільшенням кількості рядків, так і з появою різних особливостей рядків, які посилюють відмінності між ними. Загалом пошук саме такої однієї функції може бути практично недоцільним.

Тим не менше, якщо розглянути увесь твір як один рядок, вважаючи його оптимальним, то описаний щойно спосіб побудови відповідної функції (класу функцій) знову дає результат. Так, кількість складів у ньому (в узагальненому ”рядку”-творі) зростає,

але сама процедура дає подібні результати (якщо знехтувати можливими однаковими тривалостями та повторами складів у більшому тексті, які призводять до того, що деякі слова можуть бути переставлені місцями, що робить текст лише приблизно оптимальним). Можна запропонувати подальше вдосконалення цієї процедури (наприклад, введення множників, які залежать від значень сусідніх тривалостей складів), що дозволяє зменшити неоптимальність тексту.

Але все одно, якщо перейти від даного тексту до іншого, отримана функція уже, скоріше всього, не буде оптимальною.

Робимо висновок, що для усієї множини доцільно організованих текстів існування універсальної функції бачиться як певна гіпотеза (з майже напевно заперечною відповіддю).

Зв'язок з фракталами

На цей принциповий "злам", принципову неподібність текстів звернемо увагу.

Але спершу зауважимо наступне. Розглянута функція вище робить оптимальним один варіант рядка, водночас можна розглядати не один варіант, а декілька, серед яких обирати найкращий. Тоді задача зводиться до пошуку оптимальної групи варіантів з можливо мінімальною кількістю складових.

Якщо знову розглянути граничий випадок групи з кількістю елементів рівною факторіалу числа слів у рядку, то легко переконатися з комбінаторних міркувань, що така група містить (складається з, є) усі варіати рядка з даних слів, серед яких має бути той, який буде результатом. Але уже для 5 слів така група містить $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$ варіантів по 5 слів, разом це 600 слів, щоб прочитати які потрібно зазвичай кілька хвилин.

Якщо зменшити кількість слів у оптимальній групі, то серед них може не бути потрібного варіанту, але час читання зменшується відповідно. Отже, якщо маємо якийсь спосіб хоча б приблизного упорядкування, то доцільним буде пошук уже оптимального співвідношення "розмір групи – ступінь влучності приблизного упорядкування".

Це призводить до ідеї застосування різних функцій, які легко обчислюються, мають певну характерну поведінку і дозволяють приблизно упорядкувати множини усіх перестановок відповідно до завдання пошуку, і пошук уже не єдиного варіанту, а оптимальної групи варіантів.

При цьому принциповими є етапи пошуку, коли програма не дає повністю бажаного результату. Тоді можна щось змінити в даних (не змінюючи самі слова), або змінити слова, щоб змінити ті варіанти, з якими працює програма, оскільки це змінює структуру множини усіх варіантів і дозволяє отримати інші все ще певною мірою "оптимальні" варіати з точки зору автора та з точки зору програми. Це дало поштовх так званому рекурсивному режиму роботи, коли зміною даних є злиття слів (і отже,

виключення з розгляду варіантів, де вони стоять не підряд), а також режим кількох варіантів.

Цікавим спостереженням також є те, що при рекурсивному режимі можна отримати рядки, в яких більш помітними є ”злами” ритмічності, які частіше всього можна включити в ритмічність через введення додаткових пауз. Відомими в теорії є також вірші з паузами (наприклад, так званими цезурами), для яких це може бути корисним.

Також можливість утворити рядки з рядків, а кожен рядок з певних складових, нагадує фрактали. Тим не менше, зв’язок з фракталами потребує більш детального та глибокого вивчення.

Зацікавлений читач може звернутися до літератури.[24, 66, 19, 55, 75, 50, 35, 71, 59, 73, 64, 33, 32, 51, 54, 67, 42, 74, 37, 78, 77, 76, 52, 4, 1]

Аналогія з екстремальним принципом

У механіці та оптиці відомим є принцип екстремальності дії (вводиться фізична величина дії, яка для реальних процесів у цих галузях приймає серед усіх можливих значень за траєкторіями мінімальне (найчастіше) або максимальне значення, тобто одним словом екстремальне значення). У термодинаміці є закон зростання ентропії замкнених систем. У випадку механіки та оптики пошук дійсних траєкторій зводиться до пошуку тих з можливих траєкторій, для яких значення функціоналу дії є екстремальним, що дозвляє скористатися апаратом вищої математики для пошуку саме цих екстремальних значень.[28]

Аналогією до цього принципу є порівняльний режим роботи у поєднанні з режимом кількох властивостей.

Можливість використовувати власні тривалості репрезентацій звуків чи фонетичних явищ

Програми пропонують за замовчуванням чотири різні множини тривалостей фонетичних представлень. але починаючи з версії 0.13.0.0 є можливість задати власні тривалості. Для цього потрібно прописати їх як числа типу Double у файлі в порядку, який визначається наступним чином:

```
UZ ’A’ D       дз (твердий)          8
UZ ’A’ K       дз (м’який)           9
UZ ’B’ D       ж  (твердий)          10
UZ ’B’ K       ж  (пом’якшений)      11
```

```
UZ 'C' S        й                       27
UZ 'D' N        сь                      54
UZ 'E' L        ч  (твердий)            39
UZ 'E' M        ч  (пом'якшений)        40
UZ 'F' L        ш  (твердий)            41
UZ 'F' M        ш  (пом'якшений)        42
    G                                   55
    H           ю                       56
    I           я                       57
    J           є                       58
    K           ї                       59
    L           '                       60
    M           '                       61
    N           нт                      62
    O           ст                      63
    P           ть                      64
    Q           дзь                     12
    R           зь                      13
    S           нь                      65
    T           дь                      14
UZ 'a' W        а                       1
UZ 'b' D        б  (твердий)            15
UZ 'b' K        б  (пом'якшений)        16
UZ 'c' D        ц  (твердий)            38
UZ 'd' D        д  (твердий)            17
UZ 'd' K        д  (м'який)             18
UZ 'e' W        е                       2
UZ 'f' L        ф  (твердий)            43
UZ 'f' M        ф  (пом'якшений)        44
```

```
UZ 'g' D      ґ  (твердий)        19
UZ 'g' K      ґ  (пом'якшений)    20
UZ 'h' D      г  (твердий)        21
UZ 'h' K      г  (пом'якшений)    22
UZ 'i' W      і                   6
UZ 'j' D      дж (твердий)        23
UZ 'j' K      дж (м'який)         24
UZ 'k' L      к  (твердий)        45
UZ 'k' M      к  (пом'якшений)    46
UZ 'l' S      л  (твердий)        28
UZ 'l' O      л  (м'який)         29
UZ 'm' S      м  (твердий)        30
UZ 'm' O      м  (пом'якшений)    31
UZ 'n' S      н  (твердий)        32
UZ 'n' O      н  (м'який)         33
UZ 'o' W      о                   3
UZ 'p' L      п  (твердий)        47
UZ 'p' M      п  (пом'якшений)    48
UZ 'q' E      ь                   7
UZ 'r' S      р  (твердий)        34
UZ 'r' O      р  (м'який)         35
UZ 's' L      с  (твердий)        49
UZ 't' L      т  (твердий)        50
UZ 't' M      т  (м'який)         51
UZ 'u' W      у                   4
UZ 'v' S      в  (твердий)        36
UZ 'v' O      в  (пом'якшений)    37
UZ 'w' N      ць                  66
UZ 'x' L      х  (твердий)        52
```

```
UZ 'x' M        х   (пом'якшений)       53
UZ 'y' W        и                       5
UZ 'z' D        з   (твердий)           25
UZ 'z' K        з   (м'який)            26
```

де вказані значення у списку відносяться до фонетичих представлень (з модуля Languages.Phonetic.Ukrainian.Syllable.ArrInt8). Останній стовпець є восьмибітними цілими числами (GHC.Int.Int8), якими представляються ці ж звуки в нових модулях.

При бажанні задати кілька таких множин (до 9 включно), можна з нового рядка вказати '*' чи кілька таких символів, а тоді з наступного рядка буде нова множина значень.

Кожна множина має бути в такому порядку: [-1,1,2,3,4,5,6,7,8,9,10,11,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31, 32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,66]

де число відповідає останньому стовпчику в наведеній вище схемі. -1 відповідає паузі між словами (не впливає на результати пошуку рядка).

Тоді при виклику програми десь серед аргументів командного рядка (немає значення, де саме) вказати «+d» <шлях до файлу зі вказаними даними>. Програми прочитають ці значення і перетворять їх у відповідні значення. Як властивості потрібно тоді використовувати такі, які починаються з літери «H», а далі злитно з нею відповідне позначення властивості. Наприклад, «Hw04», при цьому остання цифра в записі у такому випадку означатиме порядковий номер множини значень, починаючи від 1 (максимально 9).

Поряд з власними значеннями можна використовувати і задані бібліотечно, як звичайно, в режимі кількох властивостей.

Мінімальна граматика для можливого збереження смислу та зрозумілості

Програми використовують перестановки слів, які нехтують будь-якими (чи принаймні частиною) граматичних зв'язків, порядком слів тощо. Це може призвести (крім необхідності вдумування) до ситуацій коли граматично пов'язані конструкції мови розриваються, їх частини переносяться в інші місця, утворюючи нові зв'язки і змінюючи смисл тексту.

Щоб цього було менше, щоб усунути деякі з таких ефектів, програми застосовують конкатенацію слів, які мають тісний граматичний зв'язок, щоб не розривати їх при аналізі. Це дозволяє зберегти більшу смислову легкість та впізнаваність тексту, а також як побічний ефект збільшити загальну довжину рядка, який може бути проаналізований. В українській мові граматично пов'язані

службові чи залежні слова йдуть переважно перед незалежним чи головним, тому застосовано приєднання цих службових чи залежних слів до наступного за ними. Навпаки, для часток «б» / «би» та «ж» / «же» було прийнято, що вони частіше вживаються після слова, до якого відносяться, тому вони приєднуються до попереднього слова. Повнота визначення таких випадків не є вичерпною, але розглягнуті найбільш часті випадки.

Для загального випадку потрібно мати на увазі, що службові чи залежні слова можуть йти після незалежного чи головного, тому це потрібно розглядати окремо (і приєднувати такі слова до попереднього, а не до наступного). Наразі в узагальненій версії phonetic-languages-simplified-generalized-examples-array реалізовано обидва варіанти, починаючи з версії пакету 0.15.0.0.

Зменшена множина перестановок як варіант універсальної

За замовчуванням програми аналізують універсальну множину перестановок усіх слів та їх конкатенацій, при цьому кількість аналізованих варіантів зростає як факторіал числа таких слів чи сполучень. Текст, організований більш чи менш узгоджено щодо тієї чи іншої властивості, може кардинально відрізнятися від початкового, що ускладнює розуміння та має ефект затримки обчислень.

Для експрес-перевірки можливого покращення тексту з використанням підходу введено множини перестановок лише одного слова відносно тексту, а також двох слів чи їх сполучень як універсальні множини (додатково до повної). При роботі програм вони задаються параметром командного рядка «+p» десь серед аргументів (положення не має значення). У такому випадку використовуються зменшена множина перестановок. Кількість слів та їх сполучень, які можуть розглядатися програмою як один рядок для аналізу збільшується у такому випадку до 10. Щоб використати мінімально можливу множину перестановок, потрібно задати як наступний аргумент командного рядка ”1” (лише одне слово), для попарних перестановок – ”2” (парна перестановка). Тоді серед аргументів командного рядка буде вираз ”+p 1” у першому випадку, ”+p 2” – у другому, а для повної множини перестановок, як і раніше, можна не задавати цю групу (повна множина, таким чином, використовується за замовчуванням).

У цьому випадку аналіз відбувається значно швидше (бо значно зменшується кількість розглядуваних випадків), і текст змінюється менше, що дозволяє зберегти його більшу впізнаваність. Можливе послідовне застосування цього випадку, проте слід мати на увазі, що у випадку попарних перестановок так можна отримати не найкращий варіант з точки зору підходу, також це може зайняти у такому випадку навіть більше часу, ніж аналіз усієї множини перестановок (бо деякі варіанти будуть аналізуватися по кілька разів, чого не відбувається у першому випадку). У випадку перестановок лише 1 слова відносно тексту виглядає так, що якщо при послідовному застосуванні вибору максимального елемента двічі підряд за тими ж властивостями

отримуємо один і той же варіант, то шанси , що він і буде максимальним для повної множини перестановок, збільшуються в порівнянні з випадком попарних перестановок, але оцінити збільшення (якщо ця гіпотеза дійсно вірна) складно у загальному випадку. Все ж на практиці пошук такого ”мінімально покращеного” варіанту є перспективним, оскільки він може добре зберігати смисл, покращуючи ритмічну структуру. При цьому слід враховувати наступне: при пошуку максимального елемента за значенням властивості (тобто без зміни структури) якщо при аналізі зменшеної множини перестановок отриманий текст, який співпадає з початковим, то є хороші шанси, що саме цей варіант є оптимальним з точки зору розглядуваної властивості (хоча це не гарантується). І ще одне: у такому випадку досягається локальний максимум (який може бути глобальним, а може і ні). Якщо ж повторне застосування призводить до утворення іншого (уже третього) варіанту, значить, попередній локальний максимум точно не був глобальним і програма рухається в його напрямку.

Дивіться також: [3, 20, 22, 2, 25, 27, 21, 26, 10, 5, 11, 12, 17, 13, 14, 16, 15, 6, 7, 8, 9]

Більше про ритм у музиці та мові

У роботі [49] розглядається музичний ритм, метр, пульс та інші подібні теми.

Під пульсом розуміється один з серій регулярно повторюваих однакових стимулів (звуків). Під метром розуміється міра кількості пульсів між більш чи менш регулярними акцентами. Акцентами названі ті пульси, які для людини видаються виділеними серед інших. Під ритмом (музичним) у роботі розуміється те, як один чи кілька неакцентованих бітів (пульсів у метричному контексті) групуються навколо одного акцентованого. Акцентованість може виникати по-різному і не співпадає з підсиленнями звуку, а може бути викликана найрізноманітнішими факторами в музиці.

Зазначається, що немає строгих і простих правил визначення ритмічних груп, але є певні тенденції (принаймні у Західній музиці – *Прим. автора*), на які потрібно зважати. Іноді також групування може бути двозначним (багатозначним), поліваріантним, допускати водночас різні варіанти, які співіснують разом.

Групування є архітектонічним, тобто має різні рівні, які взаємопов’язані між собою. На різних рівнях групування є продуктом подібності та відмінності звуків, також близькості та роздільності, які відчуваються органами чуттів і організовуються розумово.

Тут зазначу, що програми розглядають структури з однаковою кількістю складів в одній групі, але виділеними можуть бути у групі не один елемент (склад), а кілька, ледве чи не всі. Більше того, розглядається ступінь виділеності, який розглядається як міра значимості тих чи інших складів при обчисленні значень властивостей. Ці групи складів можна співвіднести з метром, тоді як різна кількість значимих складів відповідає тому факту, що метр може існувати безвідносно ритму. У класичному розгляну-

тому підході ритмічною є група з одним акцентованим звуком. У програмах немає чіткої залежності між акцентованим складом і мірою значимості. Це означає, що пропонується інший підхід до цього питання, результати якого можуть бути плідними. Тим не менше, можна часто вважати, що групі значимих складів відповідає ритмічна група на одному з архітектонічних рівнів, і тоді структурам з максимальним значенням відповідних властивостей такого типу відповідатимуть рядки з більш регулярним повторенням ритмічних груп у їх загальних рисах, тобто більша рівномірність на вищому архітектонічному рівні композиції твору, а меншим значенням відповідних властивостей такого типу відповідатимуть нерівномірно розподілені по рядку ритмічні групи, або їх видозміни, тобто більш складний (і тому, можливо, менш впізнаваний та відчутний, менш значимий) ритмічний малюнок на вищому архітектонічному рівні. Також можна орієнтовно вважати, що для максимальних за значеннями відповідних властивостей рядків ритмічне групування пов'язане з метричним, якщо ж розглядати не максимальні елементи, то міра зв'язку між ритмічним групуванням та метричним послаблюється, тобто такі рядки тяжіють (у музичному сенсі) до «вільного» ритму Східної та народної музики та «measured rhythm» григоріанських кантів [49]. Фактично це пропонує доволі широкі можливості для дослідження зв'язку між метричною та ритмічною структурою в мові.

Відмінності між ритмічністю в музиці та мові та їх культурна обумовленість

У роботі [60] показано натомість, що замість групування на основі принципів сприйняття (голосніший звук частіше починає групу, а також подовжений звук чи інтервал частіше завершує групу) для музики можливе групування різне для представників Великобританії та Японії, тобто є значно зумовлене культурою, а також патернами у мовах носіїв. У цій же роботі наведено багато посилань на роботи, у яких розглядалися раніше той чи інший підходи. Також вказані відміності між англійською (також українська ближча до англійської у такому контексті) та японською мовами, зокрема передування службових частин мови головним словам в англійській (і українській), тоді як слідують після головних у японській мові.

У роботі [53] згадуються відмінності між мовами у тривалостях складів: можна поділити мови на ті, в яких структура складів більш чи менш варіативна. Це зокрема впливає на неоднаковість тривалостей звуків у різних позиціях, на явища редукції голосних у мовах, де структура складів більш варіативна. У роботі [65] також зазначається, що, наприклад, у французькій мові є дуже помітним явище подовження тривалостей кінцевих складів, особливо у кінці фрази чи речення. Програми не враховують подібні відмінності в тривалостях складів, але їх врахування може бути задачею подальшого їх вдосконалення.

Про відмінності в ритмічності в музиці та мові йдеться в роботі [44]. У ній застосовано міждисциплінарний підхід. Зокрема аналізом мозкової активності при реакціях на неконгруентності (відступи від структури ритму) в мові показано, що вони значно

впливають також на процес розуміння значення мови (подовжують час реакції та розуміння). Також подібні результати щодо впливу ритму на здатність розуміння та мови наводяться в роботі [36].

У роботі [31] розглядаються особливості афро-американської ритмічної системи в музиці, зокрема виділяється така риса, як більш насичена ритмічна розробка менших частин творів та менша варіативність між меншими частинами одного більшого твору, а також ідея виразного ритму (експресивного), в якому на противагу до рівномірного біту Західної музики вводяться приблизний еквівалент біту та значно менший за тривалістю «атомічний» ритм, який, можна вважати (умовно), складається з нот приблизно 1/16 – 1/24 тривалостей та вносить «виразні деталі». Відзначається також явище суперпозиції різних циклічних ритмів в один загальний (добре простежується, наприклад, в Афро-Кубинській румбі), що схоже на ідею когерентності поліритмів для утворення одного ритму. Явище виразного «атомічного» ритму, його використання та характеристики показують, що є сенс вводити тривалості фонетичних явищ з високою (на перший погляд надмірною) точністю.

У роботі [62] показано, що діти віком 5-24 місяці сильніше реагують на музичні ритми, ніж мовні, також, що вони переважно викликають позитивні емоції у них, також рухову активність.

У роботі [41] розглядається, як ритми в музиці можуть викликати емоційні стани людини чи впливати на них. Зазначається, що це не автоматичний процес, що на нього впливають індивідуальні смаки, обізнаність та тренування.

У роботі [45] робиться спроба дослідити вплив мелодійних та ритмічних змін на сприйняття мелодії. Виявлено, що мелодії розгортаються повільніше (у сприйнятті), якщо у них більше змін по висоті звуків, більше несумісних змін у ритмічній структурі.

У роботі [58] пропонується підхід до визначення початку в звуковому музичному сигналі на основі аналізу також ритму.

У роботі [39] досліджується обробка мозком музичної інформації у досвідчених джаз-музикантів порівняно з некваліфікованими. Показано, що є помітні відмінності у здатності передбачити ритм. Це свідчить про можливість тренування відчуття ритму.

У роботі [34] експериментально показано, що попереднє прослуховування музики з відповідним подібним ритмом покращує здатність виявлення фонем у мовленні, а також цей ефект підсилює попередня аудіо-рухова підготовка до прослуховування мовленнєвих речень. Це може бути застосовано, наприклад, у логопедії. Також робиться припущення про обробку мовних та музичних часових (ритмічних) структур мозком за участю спільних (одних і тих же) ресурсів.

У роботі [57] досліджується зв’язок між часовою та ритмічною структурою музики на рівні ноти, пропонується спосіб, який суттєво враховує ритм, для визначення початку ноти у звуковому сигналі.

У роботі [61] експериментально показано, що якщо йдеться про осмислену фразу, то швидше сприймаються фонеми у наголошених складах, а якщо фраза складається з безглуздих слів (без лексичного значення), то суттєвої різниці у часі реакції для наголошених та ненаголошених складів немає. Робиться висновок, що наголошені склади можуть передбачатися (варіант

аперцепції), тоді як ненаголошені – ні (так пояснюється різниця у часі реакції). Такий результат наштовхує на думку, що наявність динамічних наголосів покращує швидкість осмилення тексту (для тих, хто є носієм культури з мовою, в якій є динамічні словесні наголоси).

У роботі [46] робиться певний підсумок щодо зв'язку ритмічної структури в мові та музиці, зокрема вказується очевидний факт наявності багато чого спільного у ритмічних групуваннях у мові та в музиці, водночас це не робить їх ідентичними. Для дослідження далі пропонується розглядати вплив мовної ритмічної структури на музичну. Згадується, що наявні дані частково підтверджують, що музична ритмічна структура зазнає впливу мовної. Під мовним (лінгвістичним) ритмом розуміється поєднання кількох факторів, які впливають на часову організацію мовлення. По-перше, це чергування слів та пауз, по друге, різні тривалості складів, по-третє, чергування наголошених та ненаголошених складів. Ці фактори можуть призвести до того, що мови можуть бути ритмічно подібними чи різними. Під музичним ритмічним групуванням розуміється також групування у фрази, чи є біт, чи він є періодичним, а також метрична структура. Спільним виступає сам факт групування у фрази, а відмінності переважно у періодичності. Зокрема, автори підкреслюють, що початкова та впливова гіпотеза про розподіл мов на ті, в яких наголоси приблизно рівномірно розподілені, та ті, в яких тривалості складів приблизно однакові, тобто початки складів приблизно рівномірно розподілені, не підтверджується наявними експериментальними даними. Це створює реальні перспективи використання програм для багатьох мов. Тим не менше, лінгвісти зберегли ці поняття та продовжують описувати ними відмінності в мовах. Ведуться дискусії щодо того, чи це справді значимі поняття, чи лише два кінці одного цілого, у якому всі мови займають своє місце. Важливим також є факт відсутності у звичайних умовах періодичності в мовному ритмі, на противагу до поширеного стану в музиці. Також автори висловлюють своє переконання, що вплив мовних ритмів на музичні не є універсальним, а більше характерний для періодів та подій, коли композитори намагаються підкреслити свою національну ідентичність та приналежність.

Поради щодо використання програм

Виходячи з вищесказаного, ритмічність утворених програмами варіантів може в багатьох випадках бути помітнішою, якщо читати слова в рядках без значних пауз між ними (як один фонетичний потік), не намагаючись виділяти наголоси. Це може і не бути так, але якщо отримані варіанти виглядають не дуже ритмічними, спробуйте саме такий варіант, і тоді порівняйте, зробіть висновки, чи підходить саме таке прочитання для Вашої ситуації.

Додаткова інформація

У роботі [29] пропонується метод класифікації музики, який використовує приховані марковські ланцюги, також інформацію щодо ритмічної структури, порівняний за точністю з ручною класифікацією.

У роботі [72] пропонується порівняно ефективний спосіб визначення ритмічної подібності фрагментів музики.

У роботі [43] вивчалося, чи виділяють піаністи-виконавці систематичними варіаціями виконання різні особливості ритмічної структури, метру та мелодії. Найбільш характерними виявилися виділення ритмічного групування.

У роботі [30] автори намагаються застосувати вейвлет-аналіз для отримання характеристик ритмічних структур музичних творів та мовлення, виділено їх математичні особливості, потім застосовано отримані результати для аналізу творів. Показано, що вони мають розумну перцептивну основу. Пропонується обговорення можливості застосування отриманих результатів до каскадного генерування ритму.

У роботі [40] розглядається статистичний поширений підхід до визначення мовної ритмічної структури, що заснований на вокальних (голосних та їх послідовностей) та інтервокальних (приголосних та їх послідовностей) інтервалах (відстанях, тривалостях) та їх варіативності. Показано, що це дозволяє краще провести типологізацію мов щодо ритмічної структури. Додатково розглядається до вже означених мов з приблизно однаковою періодичністю розподілу наголосів, мов з приблизно однаковою тривалістю складів, також мови з приблизно однаковим розподілом мор (мора – це склад з коротким голосним чи один короткий голосний, який відчувається як найменше ціле в мові замість звуку чи складу). Для оцінок використовуються варіанти індексів попарної варіативності (rPVI, nPVI) та / або середні квадратичні відхилення. Значна частина роботи присвячена описанню точок зору на питання, чи можлива кластеризація та типологізація мов за ритмом таким чином, чи вони усі утворюють певний континуум значень, розподіляючись в межах однієї великої групи. Дослідження продовжуються у цій галузі.

У роботі [56] показано, що замість тривалостей для визначення ритму можна використовувати звучність (sonority), що також дозволяє легше автоматизувати процес сегментації на ритмічні групи. Пізніші дослідження продовжують вивчати це питання, використовуючи, зокрема, приховані марковські ланцюги.

У роботі [48] продовжується вивчення впливу особливостей мовної ритмічної структури на музичну для видатних англійських та французьких композиторів минулого. Ведеться дискусія з цього питання.

У роботі [47] проводиться аналіз цього питання для німецьких та австрійських композиторів за проміжок часу близько 250 років, вказується, що є коливання, які можуть бути пояснені історичним та культурним, а не мовним впливом.

У роботі [63] продовжується ця ж тема у порівнянні також з італійськими композиторами.

У роботі [38] розглядаються теорії мелодійного музичного наголосу. Експериментальний розгляд підтверджує переважно

теорію Джозефа Томассена (1982). У цій моделі (теорії) найбільш акцентованими є поворотні точки в мелодії, коли зміна висоти нот відбувається в протилежних напрямках, причому більше це помітно для повороту з переходом від зростання до спадання, ніж для повороту з переходом від спадання до зростання. Але в різних випадках може мати місце мелодійний наголос іншого роду. Показано, що мелодійний наголос може бути доволі слабким фактором у ритмічній структурі музики.

У роботі [53] робиться аналіз, чи можуть слухачі визначати за ритмами музики (мову якої вони добре знають), якою мовою ця пісня. Як виявилося, у багатьох випадках можуть. Слухачі можуть використовувати подібності та відмінності в ритмічних структурах мов для визначення, якою мовою була складена пісня.

Цікава додаткова інформація щодо ритму та суміжних музичних явищ є в роботах:

1. Анна Виленская. Поп-музыка нулевых: два притопа или три прихлопа? Лекция Анны Виленской. Открытый Музыкальный Лекторий. 2021.
2. Анна Виленская. Земфира: ритмическая геометрия. Лекция Анны Виленской. Открытый Музыкальный Лекторий. 2021.
3. Adam Neely. Solving James Brown’s Rhythmic Puzzle. Adam Neely. 2021.

Передумови користування пакетом програм

Поки програми працюють для робочих станцій (десктопів, desktop, working station тощо), і немає мобільних версій.

Потрібно, щоб були встановлені та налаштовані програми мовою Haskell:

1. GHC (версії не раніше 7.10)
2. Cabal

Виконувані файли цих програм мають бути доступні для пошуку через змінну робочого середовища PATH (це типова їх установка).

Якщо є змога, встановіть за допомогою системного менеджера пакетів (програм) також важливі пакети Haskell bytestring, heaps, parallel. Якщо Ви плануєте користуватися також r-glpk-phonetic-languages-ukrainian-durations, то встановіть також мову програмування (а краще середовище розробки) R.

Якщо потрібні згадані пакети мовою Haskell не встановлено за допомогою системного менеджера, при встановленні пакетів вони будуть завантажені та встановлені автоматично, при цьому додатковий час піде також на їх компіляцію.

Ремарка щодо термінології

У раніших версіях пакетів використовувалися назви “норми” та “метрики” щодо властивостей текстів. Оскільки у значенні типовому для математики (зокрема функціонального аналізу) усі вказані властивості не є власне метриками і нормами (зокрема не виконується нерівність трикутника), то далі всюди буде використано замість слова “метрика” відповідно недвозначне “властивість”, маючи на увазі функціональне представлення останньої.

Встановлення пакету

Відкрийте командний рядок чи термінал і введіть як команди:

```
cabal update
cabal --reinstall --force-reinstalls install phonetic-languages-plus
cabal --reinstall --force-reinstalls install phonetic-languages-simplified-examples-array-0.17.0.0
```

Також додатково рекомендується встановити наступні пакети:

```
cabal --reinstall --force-reinstalls install r-glpk-phonetic-languages-ukrainian-durations
cabal --reinstall --force-reinstalls install mmsyn6ukr-array
```

(ця остання є опціональною, але корисною для озвучування і не займає багато місця).

Якщо є повідомлення про помилки, введіть замість update – v1-update, замість install –

```
--enable-split-sections --enable-split-objs --enable-library-stripping \
   --enable-executable-stripping v1-install
```

Якщо це не допомагає, очікуйте на новіші версії пакетів, у яких має бути виправлення багу зі встановленням, або зверніться до розробника на електронну пошту olexandr543@yahoo.com.

У новіших версіях cabal планується використовувати v2-* версії команд за замовчуванням, але поки є додаткові тонкощі при роботі з різними версіями пакетів, бібліотек та програм, тому рекомендовано використовувати v1-* варіанти. Зокрема, якщо після встановлення таким способом (v2*) при спробі завантажити модуль у інтерпретаторі GHCi будуть повідомлення, що відповідні модулі є у прихованому пакеті (hidden package), то запустіть інтерпретатор з прапором -package <назва пакету з відповідним модулем>.

Якщо пакети base, parallel та heaps часто можна встановити з репозиторіїв ОС, то решту пакетів рекомендується встановлювати з сервера Hackage (наведеними вище командами).

Зміни у версії 0.17.0.0

У новій версії зроблено спробу оптимізувати обчислення перетворення українського тексту на дані для аналізу. Також виправлено дублювання значень для ліній 4 та 3 для різних типів властивостей. Якщо Ви бажаєте відтворити попередні результати для лінії 4, то тепер потрібно для цього використовувати лінію 3, натомість 4 лінія потенційно має давати більш рекомендований автором результат. Також змінено правила нумерації версій, щоб з часом дозволити більш стабільні випуски та можливість відтворити результати роботи програм для тієї чи іншої версії у майбутньому.

Робота з програмою lineVariantsG3

Перевірте, щоб папка (каталог), куди cabal встановив виконувані файли програм, була доступна для пошуку в змінній середовища PATH.

Програма зараз підтримує два режими роботи:

- з однією властивістю (звичайний режим);
- з кількома (не більше п’яти різних) властивостями.

Останній використовується, якщо серед аргументів командного рядка є група, введена розділювачами +m <тип властивості1> <числові аргументи1> <тип властивості2> <числові аргументи2> <тип властивості3> <числові аргументи3> <тип властивості4> <числові аргументи4> <тип властивості5> <числові аргументи5> -m. Більше про нього у відповідному розділі (дивіться посилання вище). Робота програми у такому режимі описується далі в окремому розділі.

Для роботи у режимі однієї властивості введіть у командному рядку (чи терміналі) команду:

```
lineVariantsG3 <перший аргумент> [<WX аргумент> <чи друкувати значення властивості(ей)>
  <чи останнє слово має залишатися на своєму місці> <чи використовувати рекурсивний
    інтерактивний режим>] [<чи використовувати режим кількох джерел>]
    <числові аргументи> <тип властивості> <український текст>
```

все в один рядок, або з використанням переносу рядка в терміналі; або:

```
lineVariantsG3 <перший аргумент> [<WX аргумент> <чи друкувати значення властивості(ей)>
```

```
  <чи останнє слово має залишатися на своєму місці> <чи використовувати рекурсивний
    інтерактивний режим>] [<чи використовувати режим кількох джерел>]
    <числові аргументи> <тип властивості> <український текст>
    <десь серед аргументів як єдина група: обмеження>
```

та натисніть Enter. Додатково можна задавати інтерактивний режим, про що детальніше дивіться далі.

Якщо не задавати групи у квадратних дужках, Ви побачите щось на зразок наступного:

```
lineVariantsG3 10.0_1.2 yy2 садок вишневий коло хати хрущі над вишнями гудуть
```

(введений український текст в кінці команди – уривок з відомого вірша Тараса Григоровича Шевченка; загалом цей рядок – це введена команда)

```
 садок колохати хрущі гудуть надвишнями вишневий
```

(варіант (загалом може бути кілька таких варіантів, які утворюють одну групу, а також кілька таких груп; усі групи йдуть одна за одною зверху донизу у порядку зменшення кінцевого значення властивості), який максимізує обрану властивість для заданих інтервалів)

```
8.1506  (значення обраної властивості до застосування перетворення інтервалів)
8.1506  (значення обраної властивості після перетворення інтервалів,
          кінцеве значення властивості для цього рядка)
```

Зверніть увагу, що текст може (і переважно буде) писатися не так, як він пишеться згідно правил орфографії та пунктуації, але ви можете його прочитати та спробувати зрозуміти. Змінюючи введені перші аргументи, ви (скоріше всього) отримуватимете інші вихідні дані, те ж, вочевидь, стосується й українського тексту. Занадто довгий текст буде скорочений до об’єму, який ви змогли б зрозуміти (можливо, після згаданого раніше тренування) без надзусиль. Тут і далі вірш цитується за: [23].

Спробуйте оцінити, прочитавши варіант, наскільки він підходить.

УВАГА: Також потрібно пам’ятати, що в режимі однієї властивості числові аргументи передують позначенню властивості, а в режимі кількох властивостей (див. далі) навпаки – позначення властивості починає набір числових аргументів, які стосуються її, якщо такі є (інакше використовуються типові значення, які аналогічні пошуку максимального елемента).

Українські інформаційні повідомлення

Щоб програма при роботі виводила інформаційні повідомлення українською мовою (за замовчуванням виводить англійською), потрібно вказати як один з аргументів командного рядка “+u” десь не всередині груп опцій, наприклад, напочатку. Можна також налаштувати аліас для такого варіанту роботи програми, які і для інших варіантів, якщо Вам подобається. За детальною інформацією зверніться до документації командних оболонок.

Більш комплексне використання

Числові аргументи, якщо задані, мають наступне значення.

Перший числовий аргумент – кількість груп з однаковими максимальними значенням властивості (в порядку зменшення), які будуть виведені на екран як результат. Якщо задано більшу кількість, ніж їх є взагалі, то виводяться всі можливі результати, які задовольняють усі інші умови. Якщо не задано числові аргументи, то вважається рівним 1.

Другий числовий аргумент – кількість інтервалів, на які поділяється проміжок між мінімальним та максимальним значенням властивості для даного рядка. Якщо не задано, вважається рівним 1. Значення 0 не дає змоги іншим числовим аргументам далі змінювати результат роботи програми.

Усі наступні числові аргументи (якщо задано, інакше ніяких перестановок не відбувається) – номери інтервалів, які будуть поміняні місцями з максимальним за номером. Це дозволяє змінити структуру даних, які відображаються як результат роботи програми і побачити внутрішні (не максимальні) елементи. Наприклад, числові аргументи 2 6 1 4 (у такому порядку) означатимуть, що в ході виконання програма поверне 2 групи елементів з максимальними значеннями властивості (найбільшим і найбільшим наступним за попереднім), отриманим після перестановки інтервалів; відрізок між максимальним та мінімальним значенням властивості буде поділено на 6 рівних інтервалів, при цьому елементи, які знаходяться у першому та 4, рахуючи від мінімального (інтервал з номером 1) виведе максимальні 2 групи елементів.

Значення, які були в максимальному інтервалі, будуть переміщені в інтервал з найменшим номером серед тих, які переміщені в максимальний. Таким чином, при виводі ці значення будуть виведені найпізніше.

Параметр +l (+bl) та його використання

УВАГА: Якщо не було серед аргументів командного рядка символів +l, +bl, +i, +f, то після кожного рядка буде відображено 2 числа у квадратних дужках – первинне значення властивості (без переміщення інтервалів) і значення після переміщення. Якщо був (хоча б) один з цих (груп) символів – значення властивостей друкуватися не буде.

Також слід пам'ятати, що:

```
+bl == +b +l
```

(це просто скорочення використання відразу обох параметрів, замість 5 символів потрібно ввести лише 3).

Якщо Ви вказуєте також +f або +i, тоді цей параметр можна не вказувати (його буде застосовано автоматично), натомість, якщо бажаєте, можете замість нього задати додатково +b.

Параметр +b (+bl) та його використання

Якщо де-небудь серед аргументів командного рядка вказати аргумент у вигляді +b (або +bl), то програма збереже при виведенні та аналізі останнє слово у рядку на своєму місці – це дуже зручно, коли потрібно, маючи риму, підібрати інші слова. Якщо не вказувати, то усі слова будуть переміщуватися (при необхідності). Робота параметра реалізована фактично як додаткове обмеження (constraint), дивіться нижче. Ви можете також додаткво задавати інші обмеження.

```
+bl == +b +l
```

(це просто скорочення використання відразу обох параметрів, замість 5 символів потрібно ввести лише 3).

Про використання інших параметрів трохи згодом.

Режим кількох властивостей (+m …-m)

Якщо серед аргументів командного рядка задати групу аргументів, виділену розділювачами +m та -m, так щоб група аргументів виділена розділювачами +a та -a не була всередині цієї, і навпаки (щоб вони не перетиналися), тоді програма працюватиме в

режимі кількох властивостей. Значення властивостей виводитися на екран не будуть, натомість є можливість задати не більше чотирьох різних властивостей та до кожної з них вказати аргументи (дивіться: Більш комплексне використання). Програма тоді знайде варіанти, які задовольняють кожну з вказаних умов, а потім виведе на екран лише ті варіанти, які зустрічаються у всіх обраних і заданих властивостях з параметрами. Числові аргументи, які стоять після позначення властивості і передують наступному позначенню властивості, відносяться до цієї властивості. Якщо числові аргументи опущено, то використовуються значення за замовчуванням (фактично це еквівалентно простому пошуку максимальних значень властивості). Загалом, це комплексне використання даної програми.

Спробуйте, наприклад, задати:

```
lineVariantsG3 +m 02y 3 03y 3 y0 10 -m +bl <український текст>.
```

Інтерактивний режим (+i) та його використання

Інтерактивний режим (додаткова розширена взаємодія з користувачем, крім необхідної) вмикається і відповідно задається аргументом командного рядка “+i”, який можна ставити будь-де у рядку команди. У такому випадку програма виводить на екран не просто рядки, які задовольняють усі умови, але для кожного рядка виводить також його порядковий номер (починаючи з 1) в порядку посилення “слабкості” виконання усіх умов (чим більший номер, тим, у загальному випадку, більш імовірним є слабший прояв заданих умов, хоча це не завжди так – зокрема коли потрібно вивести лише одну групу). Після того запитує, яким є вибір користувача і очікує на номер варіанту, введений користувачем. Після чого повертає той варіант без номеру.

Виглядає приблизно так:

```
lineVariantsG3 +i +m w04 10 yy4 50 -m садок вишневий коло хати хрущі над вишнями гудуть
1       хрущі надвишнями гудуть садок вишневий колохати
2       хрущі надвишнями садок гудуть вишневий колохати
3       гудуть колохати хрущі садок надвишнями вишневий
4       колохати гудуть садок хрущі надвишнями вишневий

Please, specify the variant which you would like to become the resulting string by its number.
1
хрущі надвишнями гудуть садок вишневий колохати
```

Інтерактивний режим запису рядка у файл (+f ...)

Якщо задати серед аргументів групу з трьох у вигляді +f <шлях до файлу запису>, то у вказаний шлях, якщо є змога, буде дописано фінальний результат роботи програми, окрім того, що він як і в звичайному інтерактивному режимі буде виведений на екран. Ця група може стояти будь-де серед аргументів командного рядка виклику програми, але не повинна міститися всередині інших гру виду +a ... -a, +m ... -m тощо.

У результаті може вийти щось подібне:

```
lineVariantsG3 +f hello.txt +bl +m 02y 10 0y 10 y0 50 y2 40 -m садок вишневий коло хати хрущі над \
   вишнями гудуть
Please, check whether the line below corresponds and is consistent with the constraints you have
specified between the +a and -a options. Check also whether you have specified the ”+b” or ”+bl”
option(s). If it is inconsistent then enter further ”n”, press Enter and then run the program again
with better arguments.
If the line is consistent with your input between +a and -a then just press Enter to proceed further.

садок вишневий колохати хрущі надвишнями гудуть

1       вишневий колохати надвишнями садок хрущі гудуть
2       вишневий колохати хрущі надвишнями садок гудуть
3       колохати вишневий надвишнями садок хрущі гудуть
4       садок вишневий колохати хрущі надвишнями гудуть
5       хрущі вишневий колохати садок надвишнями гудуть
6       надвишнями садок хрущі вишневий колохати гудуть
7       хрущі надвишнями садок вишневий колохати гудуть
8       колохати надвишнями садок хрущі вишневий гудуть
9       колохати хрущі надвишнями садок вишневий гудуть
10      надвишнями садок колохати хрущі вишневий гудуть
11      хрущі садок надвишнями колохати вишневий гудуть
```

```
Please, specify the variant which you would like to become the resulting string by its number.
4
садок вишневий колохати хрущі надвишнями гудуть
```

Цей крайній рядок у виводі програми також буде дописаний (appended) до файлу зі вказаною назвою, якщо є змога для даного користувача.

Якщо бажаєте, можна запустити команду повторно з новим текстом та / або новими аргументами. Якщо буде таким же чином вказаний той же файл, тоді новий результат буде дописаний далі у той же файл. Це дає змогу, послідовно застосовуючи цю програму, писати чи переписувати тексти (наприклад, вірші).

Режим одночасних можливих варіацій тексту

Починаючи з версії 0.3.0.0 додана можливість опрацьовувати відразу кілька варіацій тексту, зокрема такі, що відрізняються синонімами, перефразуваннями тощо.

Для цього використовуйте замість простого тексту як крайні аргументи наступну спеціальну конструкцію:

```
{ <варіант1 українського тексту> / <варіант2 українського тексту> / ... /
  <варіантN українського тексту> }
```

усе в один рядок з хоча б двома варіантами всередині фігурних дужок. Ці варіанти будуть опрацьовуватися по черзі кожен зокрема в ході одного виклику програми, і Вами буде обраний один з варіантів (можливо, порожній). У кінці буде можливість обрати серед цих попередньо підготованих варіантів один-єдиний, який і буде результатом (і відповідно, наприклад, буде виведений на екран та записаний у файл, якщо це забезпечено аргументами командного рядка).

Будь ласка, пам’ятайте, що програма у такому режимі забезпечує обробку кожної з можливих комбінацій варіацій, і отже, якщо Ви вказали багато їх (наприклад, 3 варіації на одне слово та 4 на інше створять 3*4 = 12 варіацій, які будуть послідовно опрацьовані), поки Ви отримаєте кінцевий результат.

Рекурсивний режим роботи (“+r”)

Починаючи з версії 0.9.0.0 можна запускати програму в рекурсивному інтерактивному режимі. Для цього потрібно викликати команду з параметром “+r”, наприклад, на початку після першого аргументу. У такому випадку програма буде виконуватися рекурсивно, пропонуючи завершити рекурсію на кожному окремому кроці. При цьому її результатом буде останній перед завершенням результат роботи.

Цей режим несумісний з обмеженнями (тому що обмеження втрачають своє правильне значення і починають “зміщуватися” з потрібних частин тексту на інші), тому має використовуватися окремо від них, може бути їх альтернативою. При цьому при кожному виклику немає змоги змінити властивості та параметри виклику цих крайніх, тому обирайте їх доречно.

Зміни в тексті в цьому режимі задають через так званий рядок інтерпретатора, тобто текстовий ввід, який у арифметичному виразі – числі або частці кодує наступні дії програми.

- Якщо в рядку інтерпретатора введене двоцифрове число, тоді першу цифру програма намагається витлумачити як номер першого слова, до якого застосовується зміна, а другу цифру вона намагається витлумачити як кількість слів, які потрібно конкатенувати (з’єднати разом в одне довше “слово”), включаючи вказане перше. Далі програма (якщо введені дані можна так інтерпретувати без помилки) працюватиме з новоутвореним текстом. Відлік слів починається з 1. Наприклад, запис “12” означатиме, що програма з’єднає перше слово (цифра “1”) з наступними, щоб кількість з’єднаних була рівна 2 (цифра “2”), тобто з’єднає перші два слова в одне. “34” означатиме, що програма спробує з’єднати 4 послідовних слова, починаючи від 3-го. Якщо це неможливо зробити, програма виконає попередній етап повторно, запропонувавши ввести рядок інтерпретатора ще раз.

- Якщо введено трицифрове чи багатоцифрове число, тоді усі цифри, які не рівні 0, програма намагатиметься приписати номерам слів, які потрібно з’єднати у тому порядку, в якому цифри записані в рядку інтерпретатора.

- Якщо ввести цифру (не рівну 0 і меншу за кількість слів у рядку), потім після неї символ ділення ‘/’ і тоді ,ціле число зі знаком, то перша цифра до знаку ділення буде означати номер слова (починаючи з 1), яке буде розділятися на дві частини, а друге число буде кількість символів, які програма відрахує від початку цього слова (якщо число додатнє) вліво або з кінця слова вправо (якщо число від’ємне), щоб його поділити (додатній випадок схожий на функцію стандартної бібліотеки Haskell ‘splitAt’). Тоді вона, якщо вказані дані вдалося так інтерпретувати, розділить вказане слово на дві частини (одна з яких може бути порожньою, порожнім рядком) і далі працюватиме з текстом, у якому вказане слово замінено цими двома (або

лише першим з них, якщо вони непорожні і кількість слів у рядку уже була рівна 7 до моменту поділу). Наприклад, “садоквишневий” рядок інтерпретатора рівний “1/5”, якщо це перше слово у тексті, перетворить на “садок вишневий” (відрахувавши від початку 5 символів) і працюватиме з новим текстом далі.

WX аргумент

Якщо Ви серед властивостей використовуєте “w” або “x” серію (або обидві), то для них можна задати окремий аргумент, який має починатися з “+x” з наступним записом двох додатніх чисел подвійної точності (типу Double), з’єднаних підкресленням. Наприклад, +x 2.345_0.45676237876. Якщо цей аргумент не задати, то буде використаний аргумент за замовчуванням +x 2.0_0.125.

Зверніть увагу, починаючи з версії 0.16.0.0, тепер між +x та числами є пробіл, раніше його не було.

Перше число подвійної точності буде використане як множник (чи дільник у випадку неспівпадіння) і найбільше впливає на значення властивості, стосується найважливіших складів у стопі (ритмічній групі); натомість другий буде використаний або лише для збільшення значення властивості при співпадінні менш значимих довжин складів у стопі (ритмічній групі) (що відповідає “w” серії), або також і для більш складної форми поведінки (“x” серія).

Більше деталей можна побачити у розділі Типи властивостей.

Режим використання кількох джерел

Якщо серед аргументів командного рядка перед українським текстом (чи замість нього) або після нього задати групу аргументів у обрамленні

```
+t {двозначне число} ... ^t
```

то програма працюватиме з використанням режиму кількох джерел. Замість трьох крапок можна вказати аргументи.

Поведінка програми відрізняється для випадків попарних перестановок (Див. Режим попарних перестановок, задається аргументом командного рядка «+p [1 чи 2]») та повної множини перестановок.

У першому випадку після +t програма очікує одне з чисел 10, 11, 20, 21, 30, 31, 40, 41, 50, 51, 60, 61, 70, 71, 80, 81, 90, 91, де 0 як друга цифра вказує, що програма рядки з кожного джерела не буде з’єднувати в один рядок перед поділом на частини для

аналізу та перетворення, а 1 там же означає, що програма спочатку з’єднає усі рядки кожного джерела в один рядок, а потім застосує до нього поділ на частини. Перша цифра (якщо не рівна 1) означає кількість слів у кожному рядку (крім останнього) у кожному джерелі, які будуть складати нові рядки для аналізу. Якщо перша цифра рівна 1, то це еквівалентно 10 (максимальна кількість слів для цього режиму).

У другому випадку після +t програма очікує одне з чисел 20, 21, 30, 31, 40, 41, 50, 51, 60, 61, 70, 71. Значення їх аналогічне до першого випадку.

На місці трьох крапок можна вказати шляхи до файлів з українським текстом, який буде використаний для аналізу. Якщо залишити порожніми ці три крапки, то програма буде запитувати кожний новий рядок у нескінченному циклі у корисувача, а далі опрацьовуватиме його згідно усіх інших аргументів. Можна також у такому випадку лише вказати +t і двоцифрове число.

Спробуйте, наприклад, такі варіанти:

```
lineVariantsG3 +r 3 w04 +t 71 ”sadok.txt” ”other_poem.txt” ”just_text.txt” ^t
lineVariantsG3 +r 3 w04 +t 71
lineVariantsG3 +r 3 w04 +f ”file_for_saving.txt” +t 51
```

де очікується, що вказані в рамках

```
+t ... ^t
```

файли існують (програма ігнорує файли, які не вдалося прочитати).

Цей режим також несумісний з обмеженнями (+a ... -a), краще замість них застосовувати рекурсивний режим.

Зверніть увагу, що в новій версії програми (з 0.16.0.0) знак множення змінений на +, між першим аргументом та числом є пробіл.

Робота з програмою unconcatUkr

При виконанні програм lineVariantsG3, rewritePoemG3 утворюються конкатенації слів для збереження мінімальної граматики та кращого розуміння тексту. Деякі з них є типовими і їх можна легко відрізнити від звичайних (неконкатенованих) слів. Тому для прискорення редагування текстів після використання програм можна застосувати ще одну програму — unconcatUkr.

```
unconcatUkr 1 <шлях до файлу з конкатенованими сполученнями слів> [<шлях до нового файлу>]
unconcatUkr 2 <шлях до файлу з конкатенованими сполученнями слів> [<шлях до нового файлу>]
unconcatUkr -i
```

це три принципові способи використання цієї утиліти.

Перший найбільш безпечний, застосовує лише ті розділення сполучень, які майже напевно не призведуть до неправильних слів. Другий більш ризикований, але має більший ефект, тим не менше, застосовуйте з обереженістю (деякі не настільки широко вживані слова будуть неправильно розділені при цьому).

За замовчуванням, якщо цифра не вказана, то вона вважається 1.

Робота з програмою propertiesTextG3 (та distributionTextG)

І варіант (лише рядки)

Перевірте, щоб папка (каталог), куди cabal встановив виконувані файли програм, була доступна для пошуку в змінній середовища PATH.

Далі введіть у командному рядку (чи терміналі) команду:

```
propertiesTextG3 <перший аргумент> [<WX аргумент> <чи “вирощувати рядки”>] <файл з
  українським текстом> <контроль кількості інтервалів> <контроль друку також рядка
    тексту> <контроль розбивки тексту на рядки> <тип властивості>
```

та натисніть Enter.

Ви побачите щось схоже на:

```
propertiesTextG3 2.1_3.0 sadok.txt s 1 0 03z
      4
8     8    8    1.0029  1.0029  1.0000  1.00143062     2       4     Тарас ШЕВЧЕНКО
      Вказематі
108   113  244  1.0425  2.2559  2.1640  0.64036546     3       1     Садок вишневий колохати
14    20   20   1.4468  1.4468  1.0000  1.18260870     3       4     Хрущі надвишнями гудуть
0     17   46   Infinity   Infinity   2.6243  0.76210526     3     2     Плугатарі зплугами йдуть
```

```
108   108  110  1.0000  1.0182  1.0182  0.99097861     3       1     Співають ідучи дівчата
8     13   34   1.5441  4.0000  2.5905  0.61763770     3       1     Аматері вечерять ждуть
109   248  436  2.2741  4.0000  1.7589  0.90963892     3       2     Сем’я вечеря колохати
12    12   27   1.0000  2.2500  2.2500  0.61538462     3       1     Вечірня зіронька встає
0     16   16   67.4286 67.4286 1.0000  1.97077244     3       4     Дочка вечерять подає
248   508  508  2.0437  2.0437  1.0000  1.34289977     3       4     Амати хоче научати
1     13   13   24.6349 24.6349 1.0000  1.92198142     3       4     Так соловейко недає
108   112  113  1.0346  1.0438  1.0089  1.01242119     3       4     Поклала мати колохати
0     13   13   Infinity   Infinity   1.0524  1.90042674     3     4     Маленьких діточок своїх
3     4    4    1.1995  1.3353  1.1132  1.02728205     3       3     Сама заснула колойіх
108   108  248  1.0000  2.2937  2.2937  0.60721063     4       1     Затихло все тілько дівчата
3     13   13   4.0000  4.0000  1.0000  1.60000000     2       4     Тасоловейко незатих
      Міжітравня
      С-Петербург
```

Числові стовпчики мають однакове значення для обох варіантів для рядків. Відмінність полягає у тому, що у другому випадку статистика по всьому тексту має більше значення з погляду дослідника (дослідниці), ніж по кожному рядку зокрема.

I стовпчик – це мінімально можливе значення обраної властивості для заданих даних серед усіх можливих варіантів перестановок слів у рядку;

II стовпчик – це актуальне значення обраної властивості для заданих даних у рядку, те, яке реалізується у конкретно цьому варіанті рядка;

III стовпчик – це максимально можливе значення обраної властивості для заданих даних серед усіх можливих варіантів перестановок слів у рядку;

IV стовпчик – це відношення значення властивості для даного рядка та її мінімального значення для слів, з яких складається рядок; число, яке не менше 1.0;

V стовпчик – це відношення максимального значення властивості для слів даного рядка та її мінімального значення, з яких складається рядок; число, яке не менше 1.0 і не менше числа у IV стовпчику;

VI стовпчик – це відношення максимального значення властивості для даного рядка та її актуального значення; число, яке не менше 1.0 і не більше значення у V стовпчику;

VII стовпчик – це відношення актуального значення властивості до середнього арифметичного (півсуми) максимального та

мінімального значення для всіх можливих перестановок слів при заданих даних; число, яке відображається з повною обчисленою кількістю знаків після крапки; має важливе значення для подальшої статистики для всього тексту;

VIII стовпчик – кількість слів у рядку, деякі з яких можуть складатися з кількох з'єднаних українських слів;

IX стовпчик – номер інтервалу (починаючи з 1), до якого входить актуальне значення властивості для заданих даних;

Далі направо – якщо вказано <контроль друку також рядка тексту> як "1", тоді тут виводиться рядок тексту, який аналізується; інакше, ці дані не виводяться.

II варіант – статистика по всьому тексту (+ можливо, рядкова)

Перевірте, щоб папка (каталог), куди cabal встановив виконувані файли програм, була доступна для пошуку у змінній середовища PATH.

Далі введіть у терміналі команду:

```
propertiesTextG3 <перший аргумент> [<WX аргумент> <чи "вирощувати рядки">] <файл з
  українським текстом> <контроль кількості інтервалів> <контроль друку також рядка
    тексту> <контроль розбивки тексту на рядки> <тип властивості> | distributionTextG
      <той же аргумент щодо кількості рядків> <чи виводити на екран також порядкові
        дані>
```

та натисніть Enter. У Unix-подібних ОС вертикальна лінія, виділена червоним, слугує для створення пайплайнів (pipelines) у терміналі в shell; для ОС Windows :

```
PowerShell -Command "propertiesTextG3 <перший аргумент> [<WX аргумент> <чи
  "вирощувати рядки">] <файл з українським текстом> <контроль кількості інтервалів>
    <контроль друку також рядка тексту> <контроль розбивки тексту на рядки> <тип
      властивості> | distributionTextG <той же аргумент щодо кількості рядків>
        <чи виводити на екран також порядкові дані>"
```

Ви побачите щось схоже на:

```
propertiesTextG3 2.1_3.0 sadok.txt s 1 0 02y | distributionTextG s 1 +W
        4
3       3       5       1.0000  1.7000  1.7000  0.74074074    2    1    Тарас ШЕВЧЕНКО
 Вказематі
108     108     142     1.0000  1.3111  1.3111  0.86538462    3    1    Садок вишневий колохати
7       89      89      13.2202 13.2202 1.0000  1.85935498    3    4    Хрущі надвишнями гудуть
3       25      34      8.1680  11.1680 1.3673  1.34253780    3    3    Плугатарі зплугами йдуть
108     108     142     1.0000  1.3111  1.3111  0.86538462    3    1    Співають ідучи дівчата
4       25      34      6.2298  8.5180  1.3673  1.30906760    3    3    Аматері вечерять ждуть
108     243     277     2.2500  2.5611  1.1383  1.26365055    3    4    Сем’я вечеря колохати
3       34      34      11.1680 11.1680 1.0000  1.83563445    3    4    Вечірня зіронька встає
3       89      89      29.6720 29.6720 1.0000  1.93479395    3    4    Дочка вечерять подає
108     277     277     2.5611  2.5611  1.0000  1.43837754    3    4    Амати хоче научати
3       3       13      1.0000  4.3111  4.3111  0.37656904    3    1    Так соловейко недає
108     142     245     1.3111  2.2694  1.7309  0.80203908    3    1    Поклала мати колохати
3       3       25      1.0000  8.1680  8.1680  0.21815009    3    1    Маленьких діточок своїх
3       12      25      4.0000  8.1680  2.0420  0.87260035    3    2    Сама заснула колойіх
27      29      242     1.0778  8.9778  8.3299  0.21603563    4    1    Затихло все тілько дівчата
4       4       12      1.0000  3.0508  3.0508  0.49372385    2    1    Тасоловейко незатих
 Міжітравня
 С-Петербург
--------------------------------------------------------------------------------------------------
1       2       3       4
8       1       2       5
50.00%  6.25%   12.50%  31.25%
1.0271+-0.5479  0       16
2       3       4       5       6       7
2       13      1       0       0       0
**************************************************************************************************
```

```
2       .       .       .
5       1       2       5
1       .       .       .
.       .       .       .
.       .       .       .
.       .       .       .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
2       0       0       0
5       1       2       5
1       0       0       0
0       0       0       0
0       0       0       0
0       0       0       0
=====================================================================================================
```

Виділення кольором не буде, тут просто згруповані різні типи даних статистики.

1. Червоний колір – Десятковий дріб з похибкою – це середнє арифметичне значення (математичне сподівання) всіх чисел у VII стовпчику статистики для рядків, плюс-мінус середнє квадратичне відхилення; у випадку обраної властивості “y0” – часто число близьке до 1.0; може містити похибку округлення викликану особливостями знаходження суми чисел з плаваючою крапкою.

   У випадку, якщо всі рядки тексту виключаються з аналізу (дивіться: пояснення для оранжевого кольору), то виводиться замість матриць та подальшої інформації лише один рядок тексту – сигнальний напис:

   “1.000+-0.000!”,

   де “	”означає символ табуляції.

   Цей напис означає, що вказаний текст не підходить для аналізу програмою, а також, що будь-які дані, які в ході роботи програми можна було б отримати з цього тексту та супутніх текстів, повинні аналізуватися так, щоб не спростувати зроблені висновки на основі усієї сукупності текстів. Простіше кажучи, у такому випадку можна не враховувати текст, бо при правильному підході до аналізу та інтерпретації даних він не повинен скасувати результати.

2. Синій колір – Натуральні числа – Матриця розподілу рядків за кількістю слів та інтервалами; елемент матриці у k-му рядку та j-му стовпчику – кількість рядків, для яких значення обраної властивості з вказаними даними потрапляє в інтервал з номером j (нумерація починається з 1), який рівний номеру стовпчика матриці, і містить у рядку k слів (слів або їх сполучень, які відображаються злитно для дотримання мінімальних правил граматики при аналізі та виведенні даних), число k лежить у діапазоні між 2 і 7 включно (рядки, які аналізуються для матриці, містять від 2 до 7 слів (або написаних злитно сполучень)). Таким чином, матриця завжди має 6 рядків, а число стовпчиків буде залежати від введених та наявних даних. Фактично ця матриця заміняє графік двовимірного розподілу даних.

   На екран виводиться двічі, одна від іншої відмежована тильдами. У першому випадку нульові значення не відображаються, замість них стоять крапки. Це елемент візуалізації даних, який дозволяє краще “побачити”, як виглядає розподіл, де числові значення відповідають “висоті” на графіку розподілу (значенню функції дискретного двовимірного розподілу). У другому випадку на місці крапок стоять відповідні значення, які всі рівні 0.

   Дані для матриці отримуються з VIII та IX стовпчиків статистики за рядками.

3. Оранжевий колір – Цілі невід’ємні числа – Загальні кількості рядків. Перше число зліва – кількість рядків, які виключаються з аналізу для матриці, оскільки в них мало даних (1 або менше слів). Рівність 0 означає, що усі рядки виведені на екран беруть участь в аналізі для утворення матриці розподілу. Число справа – загальна кількість рядків у тексті, який виводиться та аналізується (включно з тими рядками, які пораховані зліва).

4. Зелений колір – Відсотки – Розподіл загальної кількості рядків за інтервалами. Сума відсоткових значень природно рівна 100%. Номери інтервалів надписані зверху над відповідними відсотковими значеннями. Наприклад, напис у цих трьох рядках типу:

   1 2

   10 15

   40% 60%

   означає, що з усієї загальної кількості рядків, які піддаються аналізу за допомогою програми (містять достатньо даних), 40% припадає на перший інтервал (з меншим значенням властивості), а 60% – на другий (відповідно з більшим значенням властивості). Тобто таких рядків відповідно 10 і 15.

   Усі інтервали рівні за величиною, але можуть мати різні кількості рядків (яка знаходиться в ході роботи програми).

Одновимірний розподіл, за ним можна побудувати гістограму.

5. Жовтий колір – Натуральні числа – Номери інтервалів. Починається відлік з 1. Під ними відповідні кількості рядків, значення властивості для яких згідно даних потрапляє у відповідний за номером інтервал.

6. Коричневий колір – Натуральні числа – Кількість слів у рядках. Лежить в межах від 2 до 7 включно (якщо слів менше, тоді рядок отримує значення рівні 1.0 і вилучається з аналізу програмою для матриці). Під ними – відповідні значення кількості таких рядків. 0 відповідає випадку відсутності рядків з даною кількістю слів (чи сполучень, які відображаються як одне слово).

Уважне вивчення цих даних дає змогу зробити певні висновки щодо тексту, їх сукупності, самої моделі та мови.

Режим статистики за кількома властивостями (+m ... -m)

Тепер, як і для програми lineVariantsG3, можна використовувати режим кількох властивостей. Для цього замість однієї властивості можна вказати кілька в блоці виділеному розділювачами +m ... -m.

У такому разі програма виведе на екран щось на зразок:

```
propertiesTextG3 sadok.txt s 1 0 +m y0 0y 02y 03y y2 y3 yy3 -m
                                                          4
2       4       2       1       2       4       4       1       Тарас ШЕВЧЕНКО
                                                          Вказематі
3       4       4       1       2       4       2       1       Садок вишневий колохати
3       4       1       4       4       4       4       4       Хрущі надвишнями гудуть
3       4       4       4       4       4       4       3       Плугатарі зплугами йдуть
3       2       4       1       1       1       1       1       Співають ідучи дівчата
3       2       4       4       4       3       3       3       Аматері вечерять ждуть
3       1       4       3       4       2       4       4       Сем’я вечеря колохати
3       3       4       4       4       4       4       4       Вечірня зіронька встає
3       4       2       4       4       4       4       3       Дочка вечерять подає
```

```
3       4       4       4       1       4       2       1       Амати хоче научати
3       4       1       1       4       1       4       3       Так соловейко недає
3       4       4       2       1       2       4       1       Поклала мати колохати
3       1       4       1       1       1       1       1       Маленьких діточок своїх
3       1       4       2       4       1       2       4       Сама заснула колойіх
4       3       1       1       2       1       2       2       Затихло все тілько дівчата
2       4       4       1       1       1       1       1       Тасоловейко незатих
                                                                  Міжітравня
                                                                  С-Петербург
```

У такому разі не потрібно використовувати програму distributionTextG, оскільки вона не визначена для такого випадку.

Перший стовпчик (виділений кольором тут) – кількості слів у відповідних рядках; далі йдуть стовпчики згідно того порядку, як вони позначені у блоці кількох влстивостей відповідно – номери інтервалів, у які потрапляють значення відповідних властивостей. Найперше число справа (виділене червоним кольором і єдине у своєму рядку) – це кількість інтервалів для кожної властивості (вони всі однакові). Вісім стовпчиків у даному випадку до текстових записів – значить, що було 7 (= 8 – 1) заданих властивостей у блоці.

Режим “білих рядків”

Програма також переключена тепер на режим “білих рядків”, що означає, що рядки, які містять менше слів, ніж потрібно, щоб забезпечити існування хоча б двох варіантів рядка, не відображають статистику і вона не включається у загальний результат. Для цього у випадку однієї метрики та використання програми distributionTextG потрібно викликати останню з додатковим аргументом +W (означає whitelines).

Наприклад, у такому випадку Ви побачите:

```
propertiesTextG3 sadok.txt s 1 0 03y +b | distributionTextG s 1 +W
        4
        Тарас ШЕВЧЕНКО
        Вказематі
```

```
52   52   81   1.0000  1.5577  1.5577  0.78195489     3     1     Садок вишневий колохати
3    42   42   15.1056 15.1056 1.0000  1.87581959     3     4     Хрущі надвишнями гудуть
11   11   14   1.0000  1.2669  1.2669  0.88226060     3     1     Плугатарі зплугами йдуть
36   36   40   1.0000  1.1111  1.1111  0.94736842     3     1     Співають ідучи дівчата
4    11   11   2.8100  2.8100  1.0000  1.47506562     3     4     Аматері вечерять ждуть
52   52   52   1.0000  1.0000  1.0000  1.00000000     3     2     Сем’я вечеря колохати
14   14   14   1.0000  1.0000  1.0000  1.00000000     3     2     Вечірня зіронька встає
4    42   42   10.4900 10.4900 1.0000  1.82593560     3     4     Дочка вечерять подає
37   37   37   1.0000  1.0000  1.0000  1.00000000     3     2     Амати хоче научати
1    4    4    4.0000  4.0000  1.0000  1.60000000     3     4     Так соловейко недає
36   40   40   1.1111  1.1111  1.0000  1.05263158     3     4     Поклала мати колохати
1    1    14   1.0000  14.2400 14.2400 0.13123360     3     1     Маленьких діточок своїх
1    4    4    4.4444  4.4444  1.0000  1.63265306     3     4     Сама заснула колойіх
9    52   52   5.7778  5.7778  1.0000  1.70491803     4     4     Затихло все тілько дівчата
       Тасоловейко незатих
       Міжітравня
       С-Петербург
-------------------------------------------------------------------------------------------
1       2       3       4
4       3       0       7
28.57%  21.43%  0.00%   50.00%
1.2078+-0.4743  0       14
2       3       4       5       6       7
0       13      1       0       0       0
*******************************************************************************************
.       .       .       .
4       3       .       6
.       .       .       1
.       .       .       .
```

```
.        .        .        .
.        .        .        .
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
0        0        0        0
4        3        0        6
0        0        0        1
0        0        0        0
0        0        0        0
0        0        0        0
============================================================================================================
```

“Білі” рядки тут показані як відступи без статистики там, де інакше вона була б.

Режим зменшеної множини перестановок

У режимі зменшеної множини перестановок можуть працювати обидві програми propertiesTextG3 та distributionTextG. Тоді потрібно задавати в першій з них як аргумент командного рядка «+p 1 чи 2», а в другій достатньо лише «+p». У такому випадку статистика виводиться так, наче кількість слів може бути від 2 до 10 включно.

Режим статистики з фіксованим закінченням рядка (+b)

Якщо задати як один з аргументів командного рядка для propertiesTextG3 символи +b, то програма буде обчислювати усю статистику, наче останнє слово фіксоване обмеженням і не рухається. Фактично у такому випадку смисл цього символу (аргументу) є аналогічним як і для програми lineVariantsG3.

Потрібно пам’ятати, що це звужує інтервал допустимих значень властивостей і при незмінній кількості рядків змінює розподіл всередині інтервалів.

Контроль кількості інтервалів

Можливі три випадки:

- “s” – кількість інтервалів буде визначена за відомим правилом Стерджеса, де кількість випробувань буде рівною результуючій кількості рядків;
- “l” – кількість інтервалів буде визначена за рекомендацією В. П. Левинського (див.: Опря А. Т. Статистика (модульний варіант з програмованою формою контролю знань). – Навч. посіб. – К.: Центр учбової літератури, 2012. – 448 с. ISBN 978-611-01-0266-7. С. 60);
- число – кількістю інтервалів буде задане натуральне число (має бути більше 1, хоча це не перевіряється);
- щось інше – буде використано 9.

Контроль друку також рядка тексту

Якщо цей аргумент рівний 1, то праворуч від числових даних порядкової статистики буде виведено на екран також рядок, який аналізується (в уже перетвореному вигляді для аналізу). Інакше рядок не буде виведений.

Контроль розбивки тексту на рядки

Якщо задати тут 1, то текст буде спочатку згрупований в один рядок, а потім розбитий на рядки методом ділення навпіл (за кількістю слів чи їх сполучень) доти, поки довжина усіх рядків не буде менша за 8 слів чи їх сполучень. Сторонні символи при цьому будуть відфільтровані. Якщо задати 0, то текст буде аналізуватися після фільтрації сторонніх символів (приблизно) у тих рядках, які були спочатку.

Чи використовувати “вирощування рядків”

Якщо серед аргументів командного рядка задати “+g ab”, де a, b – деякі цифри, крім 0, то буде використаний режим “вирощування рядків”. Це означає, що текст буде перетворений таким чином, щоб спочатку максимальна кількість слів у рядках була не більша за другу цифру, а потім рядки згруповані так, щоб кількість слів у рядку була близькою до першої цифри, якщо цього можна досягнути, об’єднуючи рядки в один послідовно. Іншими словами відбудеться перегрупування тексту за рядками так, щоб зробити кількість слів у кожному з них ближчою до першої цифри і не більшою за 7 (для останнього використовується остача від ділення на 7). У такому випадку контроль розбивки тексту на рядки не грає великої ролі, але може перегрупувати рядки, якщо він рівний 1. Наприклад, “+g 73” як аргумент командного рядка означатиме, що після застоування розбивки тексту на рядки усі рядки буде поділено так, щоб у кожному було не більше 3 слів чи їх поєднань, а потім об’єднано рядки так, щоб у кожному було число слів близьке до 7 (не більше).

Зверніть увагу, що, починаючи з версії 0.16.0.0 між +g та двоцифровим числом стоїть пробіл. Раніше його не було.

Той же аргумент щодо кількості рядків

Мається на увазі тут має стояти те ж значення, що і на місці контролю кількості інтервалів.

Чи виводити на екран також і порядкові дані

Тут потрібно поставити 1, щоб програма надрукувала всю статистику (спочатку порядкову, а потім загальну по тексту), інакше буде надруковано лише загальну по тексту.

Вибірковий аналіз тексту за рядками

Якщо Ви виконаєте команду

```
propertiesTextG3 <шлях до файлу з українським текстом для аналізу> @n
```

то на екран буде виведений текст з файлу, який буде аналізований з номерами усіх рядків зліва від самих рядків, що відділені від тексту символом табуляції (відображається у вигляді пробілу з непостійною шириною, яка залежить від налаштувань системи).

Тоді Ви можете виконувати цю ж програму (можна і без того, але можете вказати інші номери, ніж вважатиме програма) для аналізу вибраних рядків. Для цього до команд propertiesTextG3, крім останньої згаданої, будь-де у рядку команди до вертикальної риски (до пайплайну) додайте номери першого та останнього рядка, відділені символом двокрапки (без будь-яких інших символів, зокрема без пробілів). Можна задавати декілька таких пар, інформація буде виведена у такому ж порядку. Якщо деякі номери рядків будуть зустрічатися кілька разів, вони будуть виведені (якщо вказана така опція) та проаналізовані так само кілька разів. Якщо задано контроль розбивки тексту на рядки, що рівний 1, тоді програма об’єднає і проаналізує ті рядки, номери яких були вказані і відповідають номерам при виводі команди з @n.

Це все дозволяє сфокусовано аналізувати текст чи лише його частини.

Робота з програмою rewritePoemG3

Перевірте, щоб папка (каталог), куди cabal встановив виконувані файли програм, була доступна для пошуку в змінній середовища PATH.

Програма rewritePoemG3, починаючи з версії пакету 0.12.0.0, може працювати у режимі кількох властивостей, або в режимі однієї властивості, або в порівняльному режимі.

Режим кількох властивостей (+m …-m)

Якщо серед аргументів командного рядка задати групу аргументів, виділену розділювачами +m та -m, всередині якої вказати різні властивості в їх кодуванні (див. Типи властивостей), відділяючи кожну пробілом від попередньої та наступної, а також не вказувати серед аргументів командного рядка ”+c”, то програма працюватиме в режимі кількох (можливий випадок також і однієї, якщо вказати лише її одну) властивостей.

Синтаксис роботи програми у такому режимі:

```
rewritePoemG3 <перший аргумент> <файл з українським текстом> [<чи використовувати “вирощування
  рядків”>] +m <типи властивостей> -m <числові аргументи>
```

Після успішного завершення виконання програми (не має бути жодних повідомлень) у тій же папці (каталозі), де і файл з текстом, що переписується, мають бути файли з додатковим закінченням .new.txt та префіксами, відділеними крапкою, кожний з яких відповідає введеній властивості серед вказаних. Саме у кожному з цих файлів записаний (доданий, якщо бути точними) перетворений текст (наприклад, вірш) згідно введених даних.

Введені дані стосуються усього тексту, тобто кожного рядка тексту зокрема (після його попередньої обробки програмою).

Режим однієї властивості

Якщо серед аргументів командного рядка не задавати групу аргументів, виділену розділювачами +m та -m, а також не вказувати серед аргументів командного рядка ”+c”, то програма працюватиме в режимі однієї властивості (він можливий також, якщо в режимі кількох властивостей вказати всередині групи лише одну властивість).

Синтаксис роботи програми у такому режимі:

```
rewritePoemG3 <перший аргумент> <файл з українським текстом> [<чи використовувати “вирощування
  рядків”>] <тип властивості> <числові аргументи>
```

та натисніть Enter.

Ви побачите щось на зразок наступного:

```
rewritePoemG3 10.0_1.2 sadok.txt yyy 5 1 2
```

Після успішного завершення виконання програми (не має бути жодних повідомлень) у тій же папці (каталозі), де і файл з текстом, що переписується, має бути файл з додатковим закінченням .new.txt та префіксом, відділеним крапкою, який відповідає введеній властивості. Саме там записаний перетворений текст (наприклад, вірш) згідно введених даних.

Введені дані стосуються усього тексту, тобто кожного рядка тексту зокрема (після його попередньої обробки програмою).

Більш комплексне використання

У режимі кількох властивостей та режимі однієї властивості числові аргументи мають багато в чому схоже значення, як і для програми lineVariantsG3.

Якщо вони задані, то мають наступне значення.

Перший числовий аргумент – кількість інтервалів, на які поділяється проміжок між мінімальним та максимальним значенням властивості для даного рядка. Якщо не задано, вважається рівним 1. Значення 0 не дає змоги іншим числовим аргументам далі змінювати результат роботи програми.

Усі наступні числові аргументи (якщо задано, інакше ніяких перестановок не відбувається) – номери інтервалів, які будуть поміняні місцями з максимальним за номером. Це дозволяє змінити структуру даних, які відображаються як результат роботи програми і побачити внутрішні (не максимальні) елементи. Наприклад, числові аргументи 6 1 4 (у такому порядку) означатимуть, що в ході виконання програми відрізок між максимальним та мінімальним значенням властивості буде поділено на 6 рівних інтервалів, при цьому елементи, які знаходяться у першому та 4, рахуючи від мінімального (інтервал з номером 1) буде переміщено до максимального за номером (і значеннями властивості) інтервалу, а потім записано у файл виводу результатів рядок з максимальним значенням властивості.

Значення, які були в максимальному інтервалі, будуть переміщені в інтервал з найменшим номером серед тих, які переміщені в максимальний.

Порівняльний режим роботи (+c)

Програму rewritePoemG3 можна запустити також у так званому “порівняльному” режимі, коли вона пропонує рядки (один за одним) з кількох (не більше 7) заданих файлів і записує вибраний (або порожній рядок, якщо жодний не обрано) у кінцевий файл. Так з кількох файлів шляхом їх порядкового порівняння можна створити новий. Це також дозволяє, запустивши програму двічі чи кілька разів з різними параметрами у режимі кількох властивостей, потім запускати її в порівняльному на отриманих файлах і створювати у досить невимушений спосіб їх комбінації – нові варіанти.

Ремарка: Якщо Ви плануєте отримати більше “підказок” та рекомендацій від програми, то імовірно простіше (і краще) застосувати інтерактивний режим програми lineVariantsG3 з кількома метриками, чи навіть рекурсивний режим кількох джерел тієї ж програми.

Для роботи у порівняльному режимі (comparative mode):

```
rewritePoemG3 +c <файли, відділені пробілами, з яких будуть зчитуватися рядки> <кінцевий файл>
```

Типи властивостей

Одним з принципів роботи програми є пошук серед варіантів тексту тих, для яких максимальним є значення певної функції, яка називається “властивістю” тексту (property) і становить собою певну властивість для рядків. Користувач може самостійно обрати властивість, яка буде використана при роботі програми (це робиться в командному рядку один раз протягом роботи програми заданням (або відсутністю відповідного) командного аргумента). Аргумент командного рядка виклику програми може бути:

- “y0” – найперша у часі властивість, заснована на “періодах унікальності”. Ідея полягає у тому, що оцінюються кількості звуків, або пауз, або фонетичних явищ (палаталізації приголосних), які знаходяться між послідовними появами кожного звука не в одному, а в різних словах, і шукається загальна сума таких відстаней для різних слів. Більшому значенню відповідає текст з більш плавно змінним фонетичним малюнком, (імовірно) у залежності від середньої кількості звуків у “періоді унікальності” його може бути легше чи важче промовляти; меншому значенню – навпаки – текст з більш стрімкими змінами фонетичного малюнку, можливо, з підсиленням окремих груп звуків, що більш характерно для інтонаційно виділених і / чи поетичних текстів з закликами чи підсиленими емоціями. При використанні цієї властивості (і виключно її) перший аргумент рядка виклику програми не має значення (він ігнорується програмою).

- “0y” – перша версія властивості аналізу лише ритмічності. Метрика (напівемпірична), заснована на функції ритмічності, що використовує тривалості звуків, які були синтезовані у пакеті програм mmsyn6ukr-array. Функція ритмічності натхненна античною поезією, де замість наголошених та ненаголошених складів чергувалися ритмічно короткі та довгі; також музичними долями, для яких основними є дводольний ритм та тридольний. Функція реалізована таким чином, щоб можливо простіше вловити значні викиди підритмічностей для двоскладових і трискладових випадків. Використовуючи <перший аргумент> можна змінити співвідношення цих підвластивостей і відповідно – змінити властивість.

- “02y” – подібна до “0y” властивість, яка використовує інші тривалості звуків, синтезовані завдяки пакету r-glpk-phonetic-languages-ukrainian-durations. Можливо, одна з найточніших у цій версії з наявних для задачі написання ритмічного тексту. Можна створити інші варіанти тривалостей звуків, використовуючи можливості пакету r-glpk-phonetic-languages-ukrainian-durations або іншим способом.
- “03y” – подібна до “02y” властивість, яка використовує інші тривалості звуків, синтезовані завдяки пакету r-glpk-phonetic-languages-ukrainian-durations. Можливо, одна з найточніших у цій версії з наявних для задачі написання ритмічного тексту. Можна створити інші варіанти тривалостей звуків, використовуючи можливості пакету r-glpk-phonetic-languages-ukrainian-durations або іншим способом.
- “04y” – подібна до “02y” властивість, яка використовує інші тривалості звуків, синтезовані завдяки пакету r-glpk-phonetic-languages-ukrainian-durations. Ці тривалості звуків отримані з інших даних, ніж 0y, 02y та 03y, тому обережно змішуйте їх з ними в режимі кількох метрик.
- “y” – властивість, яка обчислює властивості “y0” та “0y” у більш ефективний спосіб, ніж кожну з них поодинці, а потім перемножує отримані дані. Дає більші значення для рядків з більш плавно змінним фонетичним малюнком та такі, які більше ритмізовані (з точки зору властивості “0y”). Використання <першого коефіцієнта> внутрішньо впливає лише на підвластивість “0y”.
- “y2” – властивість, подібна до y, але використовує замість другої підвластивості (ритмічності) варіант з “02y”.
- “y3” – властивість, подібна до y, але використовує замість другої підвластивості (ритмічності) варіант з “03y”.
- “y4” – властивість, подібна до y, але використовує замість другої підвластивості (ритмічності) варіант з “04y”.
- “yy” – властивість, яка використовує властивість “y0” та “0y”, при цьому замість їх перемноження, ділить результат другої на результат першої. Максимізується для текстів з високою ритмічністю (з точки зору властивості “0y”) та групуванням однакових звуків у групи ближче один до одного. Використання <першого коефіцієнта> впливає лише на підвластивість “0y”.
- “yy2” – властивість, яка використовує властивість “y0” та “02y”, при цьому замість їх перемноження, ділить результат другої на результат першої. Максимізується для текстів з високою ритмічністю (з точки зору властивості “02y”) та групуванням

однакових звуків у групи ближче один до одного. Використання <першого коефіцієнта> впливає лише на підвластивість “02y”.

- “yy3” – властивість, яка використовує властивість “y0” та “03y”, при цьому замість їх перемноження, ділить результат другої на результат першої. Максимізується для текстів з високою ритмічністю (з точки зору властивості “03y”) та групуванням однакових звуків у групи ближче один до одного. Використання <першого коефіцієнта> впливає лише на підвластивість “03y”.
- будь-які інші варіанти даного аргумента – аналогічна до “yy” з тією відмінністю, що замість “0y” використовується “04y”. “z”-лінія
- “0z”
- “02z”
- “03z”
- “04z”
- “z”
- “z2”
- “z3”
- “z4”
- “zz”
- “zz2”
- “zz3”

- “zz4”

  Ці метрики схожі на відповідні, де z замінено на y. Але у них використовуються складніші функції ритмічності, отримані з модуля Languages.Rhythmicity.Factor пакету phonetic-languages-rhythmicity. Обережно використовуйте змішані властивості у режимі кількох властивостей, оскільки вони фактично являють собою різні підходи всередені загального методу, тому можуть давати попарно менш сумісні результати, але при правильному використанні дають гарний результат. Можливо, потрібно трохи практики, також частіше використовуйте програму propertiesTextG.

  Також при роботі з наступними властивостями використовуються ідеї поліритмічності як джерела ритму.

- ”w01” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне максимальне значення, менше – ще одне максимальне (але менше значення). Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD;

- ”w02” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне максимальне значення, менше – ще одне максимальне (але менше значення). Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD2;

- ”w03” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне максимальне значення, менше – ще одне максимальне (але менше значення). Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD3;

- ”w04” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне максимальне значення, менше – ще одне максимальне (але менше значення). Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD4;

- ”w11” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає два максимальних значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD;

- ”w12” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає два максимальних значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD2;
- ”w13” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає два максимальних значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD3;
- ”w14” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає два максимальних значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD4;
- ”w21” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще два максимальних. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD;
- ”w22” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще два максимальних. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD2;
- ”w23” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще два максимальних. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD3;
- ”w24” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще два максимальних. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD4;
- ”w31” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD;

- ”w32” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD2;
- ”w33” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD3;
- ”w34” – більш складна ритмічна структура з іншим способом визначення значення властивості, розглядає рядок як текст з ритмічними групами по 4 склади кожна, де найбільше на ритмічність впливає одне мінімальне значення, менше – ще одне мінімальне. Як варіант обчислення тривалості складів використовується бібліотечна функція syllableDurationsD4;
- ”x01” – аналогічно до “w01”, але з більш комплексною залежністю для менш значимої тривалості та, можливо, менш передбачуваними результатами;
- ”x02” – аналогічно до “w02”, але з більш комплексною залежністю для менш значимої тривалості та, можливо, менш передбачуваними результатами;
- ”x03” – аналогічно до “w03”, але з більш комплексною залежністю для менш значимої тривалості та, можливо, менш передбачуваними результатами;
- ”x04” – аналогічно до “w04”, але з більш комплексною залежністю для менш значимої тривалості та, можливо, менш передбачуваними результатами;

Наступні значення аналогічні до відповідних “w” з більш комплексною залежністю (як щойно описані вище).

До них належать:

- ”x11”
- ”x12”
- ”x13”

- ”x14”
- ”x21”
- ”x22”
- ”x23”
- ”x24”
- ”x31”
- ”x32”
- ”x33”
- ”x34”

  Якщо цей аргумент має наступний вигляд, то застосовуються поліритмічний аналіз рядка. Шукаються та перевіряються більш комплексні властивості тексту, з застосуванням складніших за структурою властивостей. Це дослідницький напрямок програм. Також є можливість задати власні налаштування, використовуючи режим “c”, “C”, “N” властивостей. Отже, наступні значення задають таке:

- ”u01” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим 1, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”u02” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим 1, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”u03” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим 1, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u04” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим 1, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”u11” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;

- ”u12” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”u13” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u14” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома, і нарешті з ще менш значимим1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”u21” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD

- ”u22” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”u23” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u24” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”u31” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”u32” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”u33” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”u34” -> Шукається поліритм з найбільш значимим 1 максимумом, потім менш значимими двома мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”u41” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”u42” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”u43” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u44” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”u51” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”u52” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”u53” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”u54” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”u61” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”u62” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”u63” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u64” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”u71” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;

- ”u72” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”u73” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”u74” -> Шукається поліритм з найбільш значимими 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 5 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”v01” та інші “v” – аналогічні до відповідних “u” ліній, з тією відмінністю, що використовують лише зростаючі варіанти функцій для визначення ритмічності. Це робить їх більш прямолінійними.

- ”s01” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;

- ”s02” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”s03” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”s04” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 масксимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”s11” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;

- ”s12” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”s13” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;

- ”s14” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 максимумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;

- ”s21” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;

- ”s22” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”s23” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s24” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”s31” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”s32” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”s33” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s34” -> Шукається поліритм з найбільш значимим 1 максимумумом, потім менш значимими 2 мінімумами, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”s41” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”s42” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”s43” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s44” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”s51” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”s52” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”s53” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s54” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 максимумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”s61” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”s62” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;

- ”s63” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s64” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 максимумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”s71” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD;
- ”s72” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD2;
- ”s73” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD3;
- ”s74” -> Шукається поліритм з найбільш значимим 2 максимумумами, потім менш значимим 1 мінімумом, і нарешті з ще менш значимим 1 мінімумом, при цьому розглядаються групи по 6 складів. Як функція для визначення тривалостей складів використовується syllableDurationsD4;
- ”t01” та інші з “t” лінії – аналогічні до відповідних з “s” лінії, з тією відмінністю, що використовуються лише збільшуючі функції для властивостей. Це робить ці властивості більш прямолінійними.
- “S” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція

  rhythmicityPolyWeightedF2;
- “T” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF20;

- “U” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF2;
- “V” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF20;
- “W” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF3;
- “X” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF30;
- “Y” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF3;
- “Z” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedF30; Наступні лінії властивостей намагаються збільшити значимість закінчення рядка і зменшити значення його початку.
- “I” лінія відповідає властивостям “W” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF2;
- “J” лінія відповідає властивостям “X” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF20;
- “K” лінія відповідає властивостям “Y” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF2;
- “L” лінія відповідає властивостям “Z” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF20;
- “O” лінія відповідає властивостям “U” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF3;

- “P” лінія відповідає властивостям “V” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF30;
- “Q” лінія відповідає властивостям “S” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF3;
- “R” лінія відповідає властивостям “T” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedEF30; Починаючи з версії 0.10.0.0 введені також наступні властивості:
- “o” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF2;
- “p” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF20;
- “q” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF2;
- “r” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF20;
- “k” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF3;
- “l” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF30;
- “m” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF3;
- “n” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLF30; Наступні лінії властивостей намагаються збільшити значимість закінчення рядка і зменшити значення його початку.

- “g” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF2;
- “h” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF20;
- “i” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF2;
- “j” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF20;
- “b” лінія відповідає властивостям “u” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF3;
- “d” лінія відповідає властивостям “v” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF30;
- “e” лінія відповідає властивостям “s” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF3;
- “f” лінія відповідає властивостям “t” лінії, з тією відмінністю, що використовується “зважена” функція rhythmicityPolyWeightedLEF30;

Режим власних налаштувань

Якщо тип починається з “c”, “A”, “B”, “C”, “D”, “E”, “F”, “M”, “N”, то програма намагається пропарсити цю властивість як закодовану конфігурацію поліритмічності. Режим для розробників та дослідників, можуть використовуватися ще складніші поліритмічні структури. Наприклад, властивість ”c114+112=2” повертає як структуру поліритмічності P1 (Ch 1 1 4) (Rhythm 1 1 2) 2, що у свою чергу означає, що будуть шукатися 1 найбільш значимий максимум, також ще один менш значимий максимум у групах з 4 складів, як функція для визначення тривалостей складів буде використано syllableDurationsD2; ”ctttff7+112111=7*3” повертає як структуру поліритмічності P2 (PolyCh [True,True,True,False,False] 7) (PolyRhythm [1,1,2,1,1,1]) 7 3, що у свою чергу означає, що

будуть шукатися 1 найбільш значимий максимум, також ще один менш значимий максимум, також 2 ще менш значимі максимуми, також один ще менш значимий мінімум, а також ще один найменш значимий мінімум у групах з 7 складів, як функція для визначення тривалостей складів буде використано syllableDurationsD3 тощо.

При використанні нових властивостей типів “A”, “B”, “C”, “D”, “E”, “F”, “M” та “N” будуть використані “зважені” функції, які намагаються врахувати також значимість розміщення частини рядка, зокрема внутрішньо будуть використані такі функції з модуля Rhythmicity.PolyRhythm як rhythmicityPolyWeightedLEF2, rhythmicityPolyWeightedEF2, rhythmicityPolyWeightedF2, rhythmicityPolyWeightedLF2, rhythmicityPolyWeightedLEF3, rhythmicityPolyWeightedLF3, rhythmicityPolyWeightedEF3 та rhythmicityPolyWeightedF3 відповідно.

Порада: якщо у виводі програми з використанням кількох метрик немає результатів (порожній вивід), збільшіть кількість груп у властивостях (хоча б для однієї) та / або додайте перетворення інтервалів, щоб змінити внутрішню структуру тих чи інших властивостей.

Перший аргумент

При ознайомленні програми можна використовувати без цього аргументу, або ввівши на його місці 1_. Надалі Ви, можливо, захочете поглибити аналіз. Тоді можна задати також перший аргумент (він йде першим у списку аргументів командного рядка, не рахуючи групу – якщо така наявна – між аргументами +a та -a) у вигляді число1_число2, де число1 та число2 – десяткові додатні дроби або цілі додатні числа, причому одне з них може бути відсутнім (тоді воно вважається рівним 1). Наприклад,

3.4_2 2.987_0.7865 0.0001_

тощо.

Тоді перше число буде коефіцієнтом, який множиться на складову властивості, яка відповідає за ритмічність двоскладових стоп, а другий – за ритмічність трискладових стоп. Відповідно вдалою комбінацією можна підкреслити, скомбінувати чи знівелювати вплив ритмічності для двоскладових чи трискладових стоп.

За відсутності цього аргументу програми поводять себе так, наче він рівний 1_1.

Обмеження (constraints)

При запуску програми lineVariantsG3 можна вказати як аргументи командного рядка обмеження. Вони дозволяють зменшити кількість обчислень, розглядати лише певні варіанти (наприклад, з певним визначеним порядком деяких слів тощо), що дозволяє фактично розширити можливості програми. Ці обмеження кодуються як аргументи командного рядка між двома спеціальними позначеннями +a та -a. Вони утворюють групу аргументів, які можуть стояти будь-де у рядку вводу даних. У залежності від того, чи вказуються ці аргументи, програма ставить чи не ставить додаткове питання для перевірки та підтвердження (так званий double check).

Типів обмежень є 6, їх можна довільно комбінувати, але з дотриманням меж для кожного з них.

На рисунку поряд видно, що усі типи реалізовані з одним аргументом, який подібний у всіх них – це кількість слів (чи їх сполучень) у рядку. Користувач, запустивши програму, уже не може коригувати в ході її роботи цю кількість, але вона важлива для обмежень загалом. Жодний з цифрових символів у обмеженнях не повинен бути більшим за це число, також це число саме не більше 6 і не менше 0. Також необхідною умовою є те, що жодні цифрові символи в межах одного кодованого обмеження не можуть повторюватися двічі. Наприклад, завідомо не є валідними обмеження: Q2235 (повтор цифр), E2 (цифрові символи там, де їх немає), T247 (7 більша 6), F0 (один символ замість необхідних двох), A37523 (7 більша 6), B5 (один символ, а мають бути ще). Неправильно задані обмеження або не вплинуть на результат (хоча буде очікуватися інше), або викличуть помилку виконання (runtime exception) і зупинку роботи програми. Оскільки результат їх застосування не є простим, тому програма при їх заданні виводить на екран рядок, до якого будуть застосовані введені обмеження з додатковим запитанням, чи всі дані введено правильно.

Типи обмежень та їх значення наведено детальніше далі.

- Обмеження E – Без вводу додаткових цифрових символів – Відповідає відсутності додаткового обмеження, в ході роботи програми не впливає на кінцевий результат.

- Обмеження Q – 4 попарно нерівні цифри в межах від 0 до кількості слів чи їх сполучень мінус 1 – Цифри – це індекси 4-х слів чи їх сполучень, взаємний порядок яких при перестановках буде збережений таким.

  Також якщо ці слова однакові (без врахування великих та малих літер), тоді це зручний спосіб зменшити об’єм даних, який буде аналізуватися.

- Обмеження T – 3 попарно нерівні цифри в межах від 0 до кількості слів чи їх сполучень мінус 1 – Цифри – це індекси 3-х слів чи їх сполучень, взаємний порядок яких при перестановках буде збережений таким.

  Також якщо ці слова однакові (без врахування великих та малих літер), тоді це зручний спосіб зменшити об’єм даних, який буде аналізуватися.

- Обмеження F – 2 попарно нерівні цифри в межах від 0 до кількості слів чи їх сполучень мінус 1 – Цифри – це індекси 2-х слів чи їх сполучень, взаємний порядок яких при перестановках буде збережений таким.

  Також якщо ці слова однакові (без врахування великих та малих літер), тоді це зручний спосіб зменшити об’єм даних, який буде аналізуватися.

- Обмеження A – 1 цифра та ще кілька попарно нерівних цифр (усі між собою нерівні) справа від неї в межах від 0 до кількості слів чи їх сполучень мінус 1 – Перша цифра – індекс елемента, відносно якого визначається розміщення усіх інших елементів (слів чи їх сполучень); усі інші цифри правіше – індекси елементів, які мають стояти в отриманих перестановках ПРАВІШЕ від елемента з індексом, рівним першій цифрі.

- Обмеження B – 1 цифра та ще кілька попарно нерівних цифр (усі між собою нерівні) справа від неї в межах від 0 до кількості слів чи їх сполучень мінус 1 – Перша цифра – індекс елемента, відносно якого визначається розміщення усіх інших елементів (слів чи їх сполучень); усі інші цифри правіше – індекси елементів, які мають стояти в отриманих перестановках ЛІВІШЕ від елемента з індексом, рівним першій цифрі.

Паралельне виконання програм

Зазвичай усі програми пакету виконуються одним ядром процесора. При цьому для всіх програм, які розглядаються, є можливість включити режим роботи на кількох ядрах – паралельні обчислення. Для цього серед аргументів командного рядка мають бути наступні:

```
+RTS -N -RTS
```

Їх розміщення не впливає на порядок та значення інших аргументів командного рядка, також між записами RTS можуть бути інші параметри. Детальніше про ці параметри дивіться документацію англійською [18].

Можна рекомендувати ці параметри лише для програми propertiesTextG3. Для інших програм вони не рекомендуються, хоча Ви можете ними користуватися (вони просто збільшать використання ресурсів).

Подяки

Автор хоче висловити подяку авторам роботи Provably Correct, Asymptotically Efficient, Higher-Order Reverse-Mode Automatic Differentiation Faustyna Krawiec, Simon Peyton-Jones, Neel Krishnaswami, Tom Ellis, Richard A. Eisenberg та Andrew Fitzgibbon за ідею щодо оптимізації, а також Mikolaj Konarski за те, що звернув увагу автора на цю статтю. Також автор висловлює подяку своїм друзям, яких він хотів порадувати цим дослідженням.

Бібліоґрафія

[1] ВД Шарко. Актуальні проблеми природничо-математичної освіти в середній і вищій школі.

[2] Наталія Костенко. ЕЛЕМЕНТИ ТОНІЧНОГО І СИЛАБІЧНОГО ВІРШУВАННЯ В ДУМОВОМУ ВІРШІ ТГ ШЕВЧЕНКА. *НАШ УКРАЇНСЬКИЙ ДІМ*, page 38.

[3] О. В. Лазер-Паньків та ін. Л. Л. Звонська, Н. В. Корольова. Ямбічна строфа // Енциклопедичний словник класичних мов., 2017.

[4] АГ Кошовий and ГІ Кошовий. Одновимірні самоподібні фрактали та їх використання у моделюванні. 2011.

[5] Під ред. Анатолія Волкова. Айрен // Лексикон загального та порівняльного літературознавства., 2001.

[6] Під ред. Анатолія Волкова. Баяті // Лексикон загального та порівняльного літературознавства., 2001.

[7] Під ред. Анатолія Волкова. Буриме // Лексикон загального та порівняльного літературознавства., 2001.

[8] Під ред. Анатолія Волкова. Варіативність // Лексикон загального та порівняльного літературознавства., 2001.

[9] Під ред. Анатолія Волкова. Верлібр // Лексикон загального та порівняльного літературознавства., 2001.

[10] Під ред. Анатолія Волкова. Адекватний переклад // Лексикон загального та порівняльного літературознавства., 2001.

[11] Під ред. Анатолія Волкова. Александрійський вірш // Лексикон загального та порівняльного літературознавства., 2001.

[12] Під ред. Анатолія Волкова. Алкеєва строфа // Лексикон загального та порівняльного літературознавства., 2001.

[13] Під ред. Анатолія Волкова. Античні розміри // Лексикон загального та порівняльного літературознавства., 2001.

[14] Під ред. Анатолія Волкова. Античні строфи // Лексикон загального та порівняльного літературознавства., 2001.

[15] Під ред. Анатолія Волкова. Асклепіадова строфа // Лексикон загального та порівняльного літературознавства., 2001.

[16] Під ред. Анатолія Волкова. Аруз, або Аруд // Лексикон загального та порівняльного літературознавства., 2001.

[17] Під ред. Анатолія Волкова. Алкманова або Архілохова строфа // Лексикон загального та порівняльного літературознавства., 2001.

[18] Glasgow haskell compiler user's guide. 7.5. using smp parallelism. [Електронний ресурс]. Режим доступу: https://downloads.haskell.org/ghc/latest/docs/users_guide.pdf. Перевірено 10 листопада 2020 р.

[19] Елена Юрьевна Муратова. Синергетический подход к проблеме смыслопорождения в системе поэтического текста. 2009.

[20] Смаглій Г. А. Теорія музики : Підруч. для навч. закл. освіти, культури і мистецтв., 2013.

[21] МИКОЛА ВАСИЛЬОВИЧ Гуцуляк. *Українське віршування 30-80-х рр. XVII ст*. PhD thesis, «Теорія літератури». Чернівці: ЧНУ ім. Юрія Федьковича, 2017, 2017.

[22] Оксана Валентинівна Кудряшова. Functional poetics. 2016.

[23] Тарас Шевченко. *Садок вишневий коло хати*. Strelbytskyy Multimedia Publishing, 2018.

[24] ЕЮ Муратова. Специфика синергетического анализа поэтического текста. 2012.

[25] К Паладян. Початки силабо-тонічної версифікації в румунській літературі. *Питання літературознавства*, (81):164–172, 2010.

[26] ОВ Любімова. Відтворення античних розмірів в українській поезії 80-х–90-х років XIX століття. *Науковий вісник Чернівецького університету. Романо-слов'янський дискурс*, (565):190–193, 2011.

[27] П Івончак. Український силабо-тонічний вірш 50-х років XIX століття. *Науковий вісник Чернівецького національного університету. Слов'янська філологія*, (585-586):80–84, 2012.

[28] Huang Wan-Li. The extremity laws of hydro-thermodynamics. *Applied Mathematics and Mechanics*, 4(4):501–510, 1983.

[29] Xi Shao, Changsheng Xu, and Mohan S Kankanhalli. Unsupervised classification of music genre using hidden markov model. In *2004 IEEE International Conference on Multimedia and Expo (ICME)(IEEE Cat. No. 04TH8763)*, volume 3, pages 2023–2026. IEEE, 2004.

[30] Neil P McAngus Todd. Segmentation and stress in the rhythmic structure of music and speech: A wavelet model. *The Journal of the Acoustical Society of America*, 93(4):2363–2363, 1993.

[31] Vijay Iyer, Jeff Bilmes, Matt Wright, and David Wessel. A novel representation for rhythmic structure. In *Proceedings of the 23rd International Computer Music Conference*, pages 97–100. Citeseer, 1997.

[32] Andrés E Coca, Gerard O Tost, and Zhao Liang. Controlling chaotic melodies. *Proc. Encuentro Nacional de Investigación en Posgrados (ENIP)*, 2009.

[33] EJ Garba. Music programming–rule-based randomization of melodic patterns. 2008.

[34] Nia Cason, Corine Astésano, and Daniele Schön. Bridging music and speech rhythm: Rhythmic priming and audio–motor training affect speech perception. *Acta psychologica*, 155:43–50, 2015.

[35] Kerri Welch. *A fractal topology of time: Implications for consciousness and cosmology*. California Institute of Integral Studies, 2010.

[36] Katie Overy. Dyslexia and music: From timing deficits to musical intervention. *Annals of the New York academy of sciences*, 999(1):497–505, 2003.

[37] David Avnir, Ofer Biham, Daniel Lidar, and Ofer Malcai. Is the geometry of nature fractal? *Science*, 279(5347):39–40, 1998.

[38] David Huron and Matthew Royal. What is melodic accent? converging evidence from musical practice. *Music Perception*, 13(4):489–516, 1996.

[39] Peter Vuust, Leif Ostergaard, Karen Johanne Pallesen, Christopher Bailey, and Andreas Roepstorff. Predictive coding of music–brain responses to rhythmic incongruity. *cortex*, 45(1):80–92, 2009.

[40] Franck Ramus. Acoustic correlates of linguistic rhythm: Perspectives. 2002.

[41] Wiebke Trost and Patrik Vuilleumier. Rhythmic entrainment as a mechanism for emotion induction by music: a neurophysiological perspective. *The emotional power of music: Multidisciplinary perspectives on musical arousal, expression, and social control*, pages 213–225, 2013.

[42] Dietmar Saupe. Algorithms for random fractals. In *The science of fractal images*, pages 71–136. Springer, 1988.

[43] Carolyn Drake and Caroline Palmer. Accent structures in music performance. *Music perception*, 10(3):343–378, 1993.

[44] Cyrille Magne, Mitsuko Aramaki, Corine Astesano, Reyna Leigh Gordon, Sølvi Ystad, Snorre Farner, Richard Kronland-Martinet, and Mireille Besson. Comparison of rhythmic processing in language and music: An interdisciplinary approach. *Journal of Music and Meaning*, 3, 2005.

[45] Marilyn G Boltz. Tempo discrimination of musical patterns: Effects due to pitch and rhythmic structure. *Perception & Psychophysics*, 60(8):1357–1373, 1998.

[46] Aniruddh D Patel. Rhythm in language and music: parallels and differences. *Annals of the New York Academy of Sciences*, 999(1):140–143, 2003.

[47] Aniruddh D Patel and Joseph R Daniele. Stress-timed vs. syllable-timed music? a comment on huron and ollen (2003). *Music Perception*, 21(2):273–276, 2003.

[48] Aniruddh D Patel and Joseph R Daniele. An empirical comparison of rhythm in language and music. *Cognition*, 87(1):B35–B45, 2003.

[49] G.W. Cooper, G. Cooper, L.B.A. MEYER, and L.B. Meyer. *The Rhythmic Structure of Music*. Phoenix books. University of Chicago Press, 1963.

[50] Jun Kigami. *Analysis on fractals*. Number 143. Cambridge University Press, 2001.

[51] Jan Andres, Jiří Fišer, Grzegorz Gabor, and Krzysztof Leśniak. Multivalued fractals. *Chaos, Solitons & Fractals*, 24(3):665–700, 2005.

[52] Jan Andres and Miposlav Rypka. Multivalued fractals and hyperfractals. *International Journal of Bifurcation and Chaos*, 22(01):1250009, 2012.

[53] Erin E Hannon. Perceiving speech rhythm in music: Listeners classify instrumental songs according to language of origin. *Cognition*, 111(3):403–409, 2009.

[54] Kevin Merges. Fractals and art. 2005.

[55] Narayan Partap and Renu Chugh. Fixed point iterative techniques–an application to fractals. *International Journal of Research in Mathematics & Computation*, 4(1):1–7, 2016.

[56] Antonio Galves, Jesus Garcia, Denise Duarte, and Charlotte Galves. Sonority as a basis for rhythmic class discrimination. In *Speech Prosody 2002, International Conference*, 2002.

[57] Norberto Degara, Antonio Pena, Matthew EP Davies, and Mark D Plumbley. Note onset detection using rhythmic structure. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5526–5529. IEEE, 2010.

[58] Norberto Degara, Matthew EP Davies, Antonio Pena, and Mark D Plumbley. Onset event decoding exploiting the rhythmic structure of polyphonic music. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1228–1239, 2011.

[59] Christopher J Bishop and Yuval Peres. *Fractals in probability and analysis*, volume 162. Cambridge University Press, 2017.

[60] John R Iversen, Aniruddh D Patel, and Kengo Ohgushi. Perception of rhythmic grouping depends on auditory experience. *The Journal of the Acoustical Society of America*, 124(4):2263–2271, 2008.

[61] Joyce L Shields, Astrid McHugh, and James G Martin. Reaction time to phoneme targets as a function of rhythmic cues in continuous speech. *Journal of Experimental Psychology*, 102(2):250, 1974.

[62] Marcel Zentner and Tuomas Eerola. Rhythmic engagement with music in infancy. *Proceedings of the National Academy of Sciences*, 107(13):5768–5773, 2010.

[63] Joseph R Daniele and Aniruddh D Patel. An empirical study of historical patterns in musical rhythm: Analysis of german & italian classical music using the npvi equation. *Music Perception: An Interdisciplinary Journal*, 31(1):10–18, 2013.

[64] Wolfgang Wildgen. Chaos, fractals and dissipative structures in language. or the end of linguistic structuralism. *Gabriel Altmann und Walter A. Koch (Hg.), Systems. New Paradigms for the Human Sciences, de Gruyter, Berlin*, pages 596–620, 1998.

[65] Corine Astésano. *Rythme et accentuation en français: invariance et variabilité stylistique*. Editions L'Harmattan, 2001.

[66] Michael F Barnsley, John E Hutchinson, and Örjan Stenflo. V-variable fractals: fractals with partial self similarity. *Advances in Mathematics*, 218(6):2051–2088, 2008.

[67] Giuseppe Vitiello. The brain is like an orchestra. better yet, it is like a jazz combo, which doesn't need a conductor. *Chaos*, 11(1):2017, 2017.

[68] Oleksandr Zhabenko. dobutokO-poetry. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/dobutokO-poetry-0.8.1.0. Перевірено 09 листопада 2020 р.

[69] Oleksandr Zhabenko. phonetic-languages-simplified-generalized-examples-array. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/phonetic-languages-simplified-generalized-examples-array. Перевірено 23 жовтня 2021 р.

[70] Oleksandr Zhabenko. phonetic-languages-rhythmicity. [Електронний ресурс]. Режим доступу: https://hackage.haskell.org/package/phonetic-languages-rhythmicity. Перевірено 24 серпня 2020 р.

[71] Alex Goldsmith. Synthesising music: exploiting self-similarity using modular forms.

[72] Andre Holzapfel and Yannis Stylianou. Rhythmic similarity of music based on dynamic periodicity warping. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2217–2220. IEEE, 2008.

[73] Rohit Sunkam Ramanujam and Bill Lin. Randomized partially-minimal routing on three-dimensional mesh networks. *IEEE Computer Architecture Letters*, 7(2):37–40, 2008.

[74] Shlomo Alexander and Raymond Orbach. Density of states on fractals:«fractons». *Journal de Physique Lettres*, 43(17):625–631, 1982.

[75] Ihor Nabytovych. ФРАКТАЛИ ТА ФРАКТАЛЬНІ СТРУКТУРИ У ХУДОЖНЬОМУ ТЕКСТІ (на прикладі прози Л. Керрола, КС Льюїса та ХЛ Борхеса). *ВІСНИК ЛЬВІВСЬКОГО УНІВЕРСИТЕТУ. Серія іноземні мови*, (18).

[76] John E Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[77] Robert S Strichartz. Fractals in the large. *Canadian Journal of Mathematics*, 50(3):638–657, 1998.

[78] Benoit B Mandelbrot and Benoit B Mandelbrot. *The fractal geometry of nature*, volume 1. WH freeman New York, 1982.