**Handwritten Text Recognition & Context Autocomplete**

# Увод

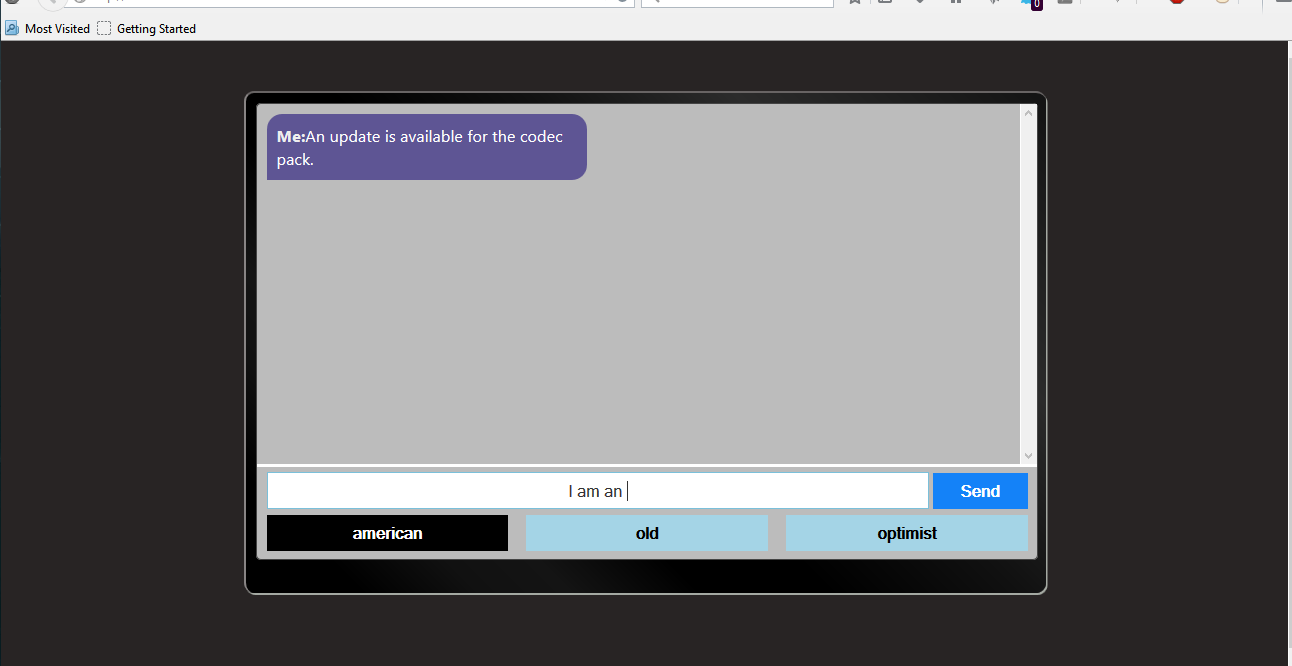
Писането на текст на ръка винаги е било най-естествено за човека. Всеки човек усвоява това умение от ранна възраст. Въпреки това изрисуването на всеки символ е сравнително по-бавно от писането на клавиатура. Съвременните софтуерни продукти предлагат опции като автоматичното довършване на думи (autocomplete), за да ускорят писането. От друга страна, когато комуникацията се осъществява чрез компютри, има затруднение при преработката на ръкописен текст до формат, разбираем за тях. Настоящият проект се опитва да улесни гореизложения сценарий като предоставя на човека възможността да изпраща съобщения изписани на ръка, но в същото време предоставя помощната функционалност за довършване на думите не само по синтактично, но и по семантично сходство.

# Общ преглед

Уеб-приложението е прототип на система за комуникация. Потребителят въвежда ръкописен текст в приложението, а алгоритъмът разпознава буквите и предлага автоматично довършване на следващата дума по смисъл. Цялостното приложение съдържа 2 оснвни модула:

* разпознаване на ръкописни букви – посредством невронни мрежи, еволюирани с генетичен алгоритъм;
* семантичен (контекстен) autocomplete – посредством n-gram модел.

Системата е персонализирана и не предлага случайни думи от речника си, а на базата на контекста, в който протича разговора – по-вероятни са думи, които отговарят на смисъла на написаното и езика на участниците в дискусията. При предоставянето от страна на потребителя на саморъчно съставен сет от тренировъчни данни (единични букви за разпознаване и текст за съставяне на контекст) се постига резултат, по-близък до индивидуалния човек.



Интерфейсът, предоставен на потребителите, приема ръчно изписани букви върху интерактивния екран. Те биват разпознати и резултатите се предават за последваща обработка на ниво думи и изрази.

Алгоритмите, реализиращи изкуствения интелект, са написани на C#.

Системата се обучава от въведения от потребителите текст, като пълно преизчисляване на n-gram вероятностите се прави веднъж дневно, иначе при получаване на нови данни се обновяват само локалните вероятности на даден gram.

# Проекти в изследваната област

Проектът е вдъхновен основно от продукта QuickType на Apple. Идеята му е да помага за по-бързото писане на текстови съобщения и имейли от телефоните им. Той надгражда над предишния софтуер, който предлага думи само на базата на речниково съвпадение.

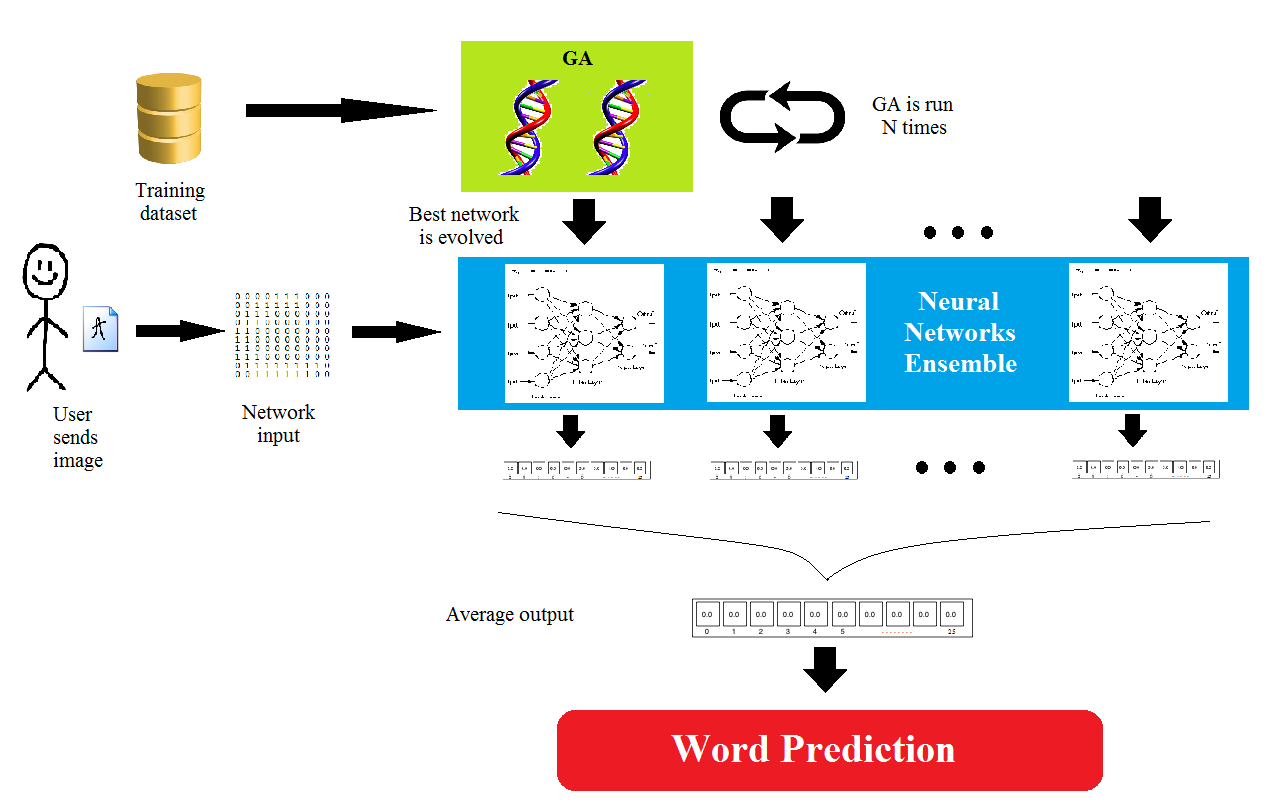
За частта с разпознаването на текст са изследвани научните трудове на различни автори, които анализират прилагането на невронни мрежи в оптическото разпознаване на текст (optical character recognition), както и използването на генетични алгоритми за намиране на възможно най-добрата мрежа, способна на това. Разгледани са способи за конвертиране на изображение до структура от данни, която може да бъде използвана като входящ сигнал за невронната мрежа – избрано е файлът да се преработва до булева матрица, която показва контура на символа. Изследвани са оптимални избори на кръстосване, мутация, терминация и други компоненти от генетичния алгоритъм. Експериментирано е с различни стойности на параметрите при тестването.

# Разпознаване На Ръкописни Букви

## Архитектура на модула

В подготвителната фаза по зададени параметри на компонентите се прилага генетичен алгоритъм за създаването на една най-добра невронна мрежа, която да се използва по време на потребителската сесия. Опционално, еволюцията може да бъде пусната няколко пъти и по този начин да се създаде цяло множество от невронни мрежи.

По време на нормалната работа на модула постъпват заявки под формата на изображения от фронтенда. Те се конвертират до входни данни за невронните мрежи. Крайният отговор е средна стойност между всички решения, излъчени от множеството невронни мрежи. Имаме ансамблова класификация.

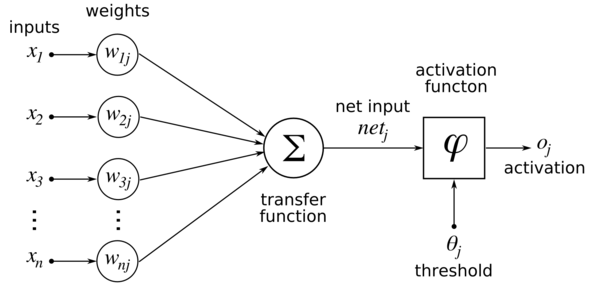


## Невронна мрежа

Невронната мрежа се състои от 1 входен, 1 изходен и произволен брой скрити слоя. Входният слой е с големина, равна на броя пиксели (широчина \* височина), от които се състои входящото изображение. Изходният слой съдържа 26 неврона и така мрежата е способна да различава 26-те главни английски букви.

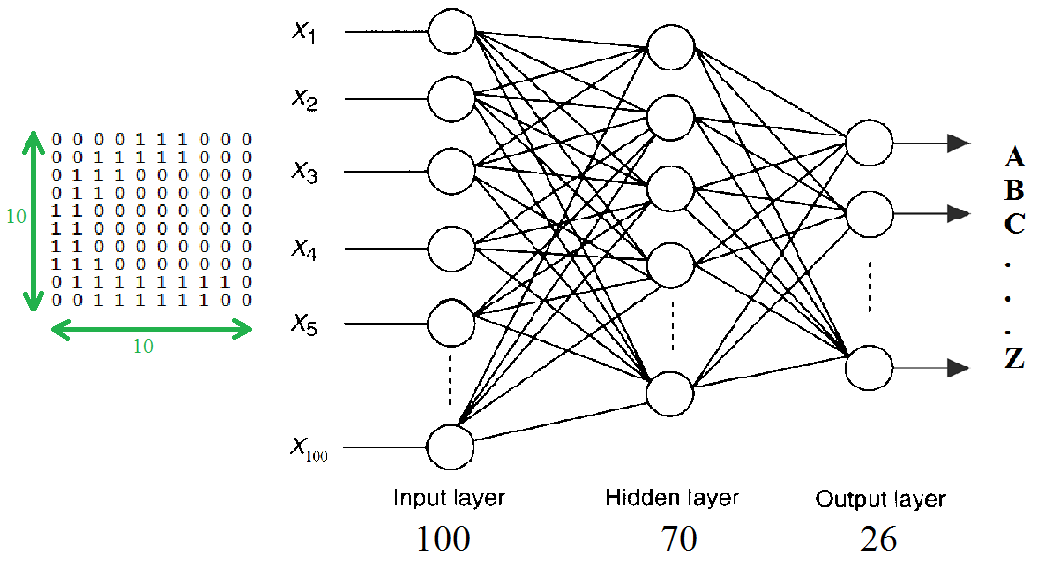
Полученият от фронтенда PNG файл съдържа 1 буква, изписана на ръка. Той се преработва, докато се получи булева матрица, чиито стойности служат за входни данни на мрежата. Изображението се изрязва – най-малкият квадрат, побиращ буквата. След това се конвертира до черно-бяла матрица – по-тъмните пиксели получават стойност 1, а по-светлите – 0.

Самата мрежа работи по познатия начин – feed-forward. Всеки неврон смята линейна комбинация на приетите сигнали (и допълнителен bias). Коефициентите са стойности в интервала [-1.0; 1.0] по подразбиране, но промяна е възможна. Сумата се предава като аргумент на сигмоидната функция, чийто резултат се излъчва като изходен сигнал. Има и реализирана функционалност за ъпдейт на теглата – back propagation, но тя не се използва от приложението, тъй като мрежите се подобряват чрез еволюция.



Мрежите имат следните конфигурации:

|  |  |
| --- | --- |
| ***Скрити слоеве*** | 1 слой: 70 неврона |
| ***Входящ слой*** | 100 неврона (всяко изображение се преобразува до 10x10 булева матрица) |
| ***Изходен слой*** | 26 неврона (по 1 за всяка главна английска буква) |
| ***Начален интервал на теглата на ребрата*** | [-1.0; 1.0] |
| ***Брой невронни мрежи, участващи в 1 ансамбъл*** | 3 |

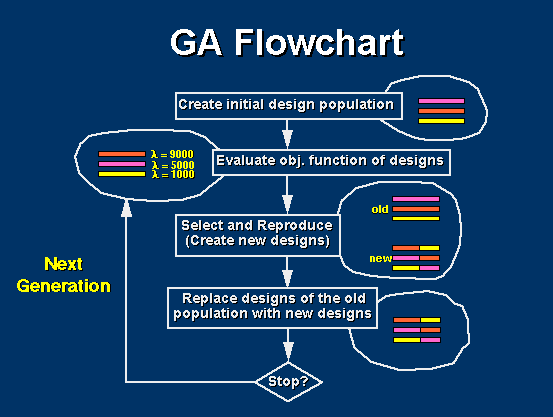


## Генетичен алгоритъм

Използвана e библиотеката GeneticSharp със свободен лиценз. Тя позволява удобна смяна на редица параметри на генетичния алгоритъм. Имплементацията използва добри образци на дизайн (design patterns), а алгоритмите са ефективно реализирани.

Следните параметри са зададени:

|  |  |
| --- | --- |
| ***Размер на популацията*** | 40 – 60 |
| ***Кръстосване*** | **Uniform** – на произволен принцип се избира ген от някой родител; приблизително половината от гените са от единия, другата половина – от другия  или  **NetworkCrossover** (собствена имплементация) – модификация на Uniform; гените не се избират поединично, ами всички тегла на даден неврон в мрежата се взимат като едно цяло – цял интервал, съответстващ на един неврон (входни тегла x1, …, xn и bias), от единия родител се взима на всяка стъпка, а не един-единствен ген |
| ***Шанс за мутация*** | 0.003 |
| ***Фитнес функция*** | Трябва да връща стойности в интервала [0.0; 1.0] (такава е имплементацията в GeneticSharp):  f = 1 / (E + 1),  където E – сумата от средноквадратичната грешка при класификацията на всички примери от тестовото множество |
| ***Терминационно условие*** | Достигане на 5000 генерации или стагнация на популацията за 50 генерации |



## Ефективност на алгоритъма и тестове

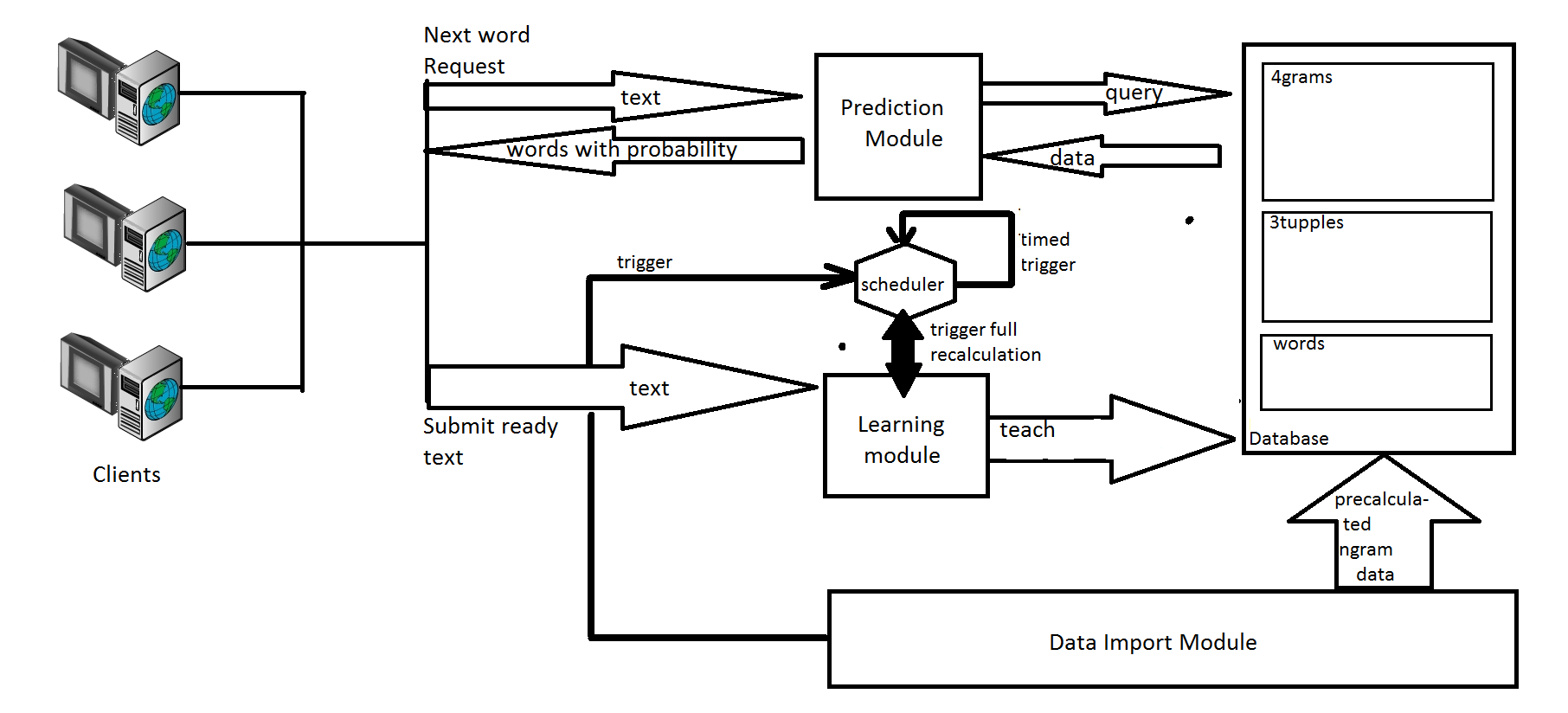
Тестването на точността се измерва по метода cross-validation. Множеството от тестови данни (класифицирани предварително примери) се разделя на K части. Всичките без една се използват за трениране, а последната – за сравнение между получените стойности и правилните отговори. Гледа се най-вече сумата от средноквадратичната грешка при класификацията на всички примери от тестовото множество. Измерва се също и процентът верни отговори – от практическа гледна точка най-полезно.

“The Chars74K dataset” съдържа множество от тестови примери (по 55 броя класифицирани изображения за всяка от главните английски букви), които са използвани тук за трениране и валидация на резултатите. Данните са конвертирани до 10x10 булеви матрици, с които работят невронните мрежи.

# Контекстен Autocomplete

## Преглед

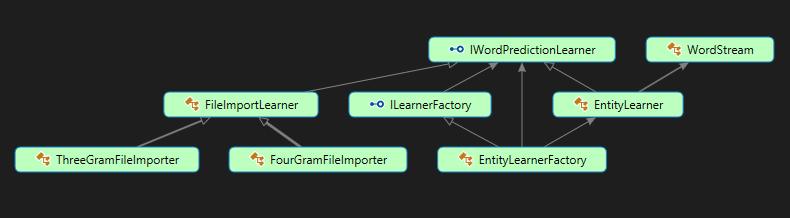
Контекстния Autocomplete е на базата на Модели на Марков, тоест предполагаме че следващата дума зависи само от краен фиксиран брой от предхождащите я, за това ползваме 4-gram модел за който задаваме вероятност за всяка наредена четворка от думи, или по точно условната вероятност P(Word4 | Word1, Word2, Word3).



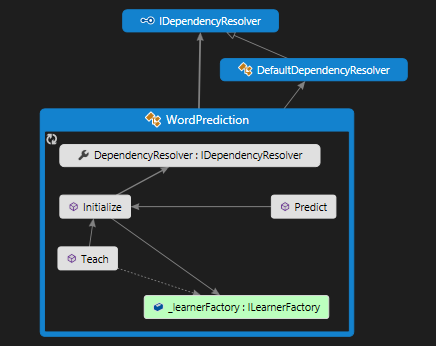
При работа на модула той обработва заявки от идващ текст и предсказва следващатите най-вероятни думи на база на предварително пресметнати условни вероятности.

## Обучение

Модула за обучение подържа обучение чрез учене от текст въведен от потребителите или n-gram-и импортнати от файлове.



Обучението на системата става чрез обекти инстанции на класове наследяващи **IWordPredictionLearner** който служи за абстрактно представяне на учител на системата.



Програмния интерфейсът който употребява клиентското приложение е на база на dependency injection , клиента трябва да предостави свой **IDependencyResolver** който да предоставя имплементации на **ILearnerFactory** интерфейса или да ползва дефоутният такъв който предоставя имплементация на **ILearnerFactory** чрез **EntityLearnerFactory** който предоставя имплементации на **IWordPredictionLearner** под формата на **EntityLearner.**

### Обучение по време на работа

Приложението MihailGospodinov.WordPrediction.Web ползва дефоутната имплементация на **IDependencyResolver** а не продоставя свой, така обучението става чрез **EntityLearner.** EntityLearner обучава системата като токенизира идващия поток от текстови данни във поток от думи ползвайки помощния **WordStream** и енумерирайки го.За всяка последователна тройка думи от потока се създава (или увеличава честотата с едно) на съответната във базата от данни. Също се прави и за последователните четворки като те биват записани във 4-gram-ове и зе изчислява тяхната вероятност по следният начин:

*P(w4 | w1,w2,w3) = (frequency(<w1,w2,w3,w4>) + 1)/( frequency(<w1,w2,w3>) + 1).*

Ползваме (+ 1) за да имплементираме така наречения +1 smoothing

### Импорт на предварително извлечени данни

При импортване на данни от файлове със предварително генерирани n-grami по текст ползваме MihailGospodinov.WordPrediction.Import приложението , което импортва данни от външен източник(файл) ползвайки ThreeGramFileImporter и FourGramFileImporter , за да импортне в базата съответно 3 и 4-грами. След приключването на импорта се преизчисляват всички вероятности чрез Stored Procedure-а изпълнявана на ръка.

## Метод на предсказване на следващата дума

Следващата дума се предсказва лесно ползвайки проста заявка която търси матч по 3 думи и връща резултатите подредени по условната им вероятност. За бързодействие на търсенето n-gram таблицата е логически индексирана по първава, втора и трета дума освен физически по ключ.

# Заключение и бъдещо развитие

В бъдеще се предвижда експериментирането с повече методи на кръстосване в генетичните алгоритми и подобрение на времето за изпълнение, което ще позволи да бъде и завишено качеството на генерираните мрежи (по-малко време за итерация означава възможност за повече итерации с по-комплексни мрежови дизайни).

Друго подобрение би било разширяване на методите на предсказване на дума до такива които ползват повече от n-gram модели и се опитват да връзват контекстно резултата с по-голяма част от предхождащия го текст.

Прототипът постига оптимистични резултати. Макар че, за да бъде полезен за всекидневна употреба като завършен продукт, изисква подобрения, приложенето илюстрира как алгоритмите от сферата на изкуствения интелект могат да бъдат приложени в решаването на всекидневен проблем. Разчитането на ръкописни букви би улеснило автора на текста, а семантичното довършване на думи и изрази според контекста на разговора би увеличило скоростта на комуникация. Завършеният продукт ще изведе софтуерните продукти за текстови съобщения на ново ниво – те ще бъдат „умни“ в помощта си към пишещия и той действително ще може да използва функционалността им за формулиране на мисълта си.

*Използвани източници*

<https://github.com/giacomelli/GeneticSharp>

<http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/>

<https://www.apple.com/ios/whats-new/quicktype/>

<http://www.imore.com/quicktype-keyboards-ios-8-explained>

<http://www.cse.unsw.edu.au/~cs9417ml/MLP2/>

<https://www.youtube.com/watch?v=aVId8KMsdUU&index=1&list=PL29C61214F2146796>

<http://www.ai-junkie.com/ann/evolved/nnt1.html>

<http://www.ics.uci.edu/~dramanan/teaching/ics273a_winter08/projects/sperez1_GANN.pdf>

<http://airccse.org/journal/ijcsea/papers/2412ijcsea01.pdf>

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.87&rep=rep1&type=pdf>

<http://homepages.inf.ed.ac.uk/pkoehn/publications/gann94.pdf>

<http://research.ijcaonline.org/volume103/number13/pxc3899258.pdf>

<http://artint.info/html/ArtInt_189.html>

<http://visualstudiomagazine.com/articles/2013/10/01/understanding-and-using-kfold.aspx>

<http://www.dotnetperls.com/fisher-yates-shuffle>

<http://www.ngrams.info/>

<https://class.coursera.org/nlp/lecture/128>

<http://www.sciencedirect.com/science/article/pii/S0957417413006271>

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.1958>