МАТЕМАТИЧКА ГИМНАЗИЈА

МАТУРСКИ РАД
из предмета
Програмирање и програмски језици

на тему

Рекурентни неуронски модел дела Иво Андрић - „На Дрини ћуприја“

Ученик:
Вук Вуковић, IV$_{\text{Б}}$

Ментор:
Петар Величковић

Београд, мај 2017.

Садржај

Глава 1

Увод

Машинско учење је област рачунарских наука која је у непрестаном развоју: од алгоритама за класификацију слика и препознавање непожељних порука у е-пошти, па све до алгоритама за аутоматску вожњу аутомобила. Самим тим, машинско учење сваким даном постаје **све применљивије у свакодневном животу**.

Мотивација за бављене овом темом потекла је од саме идеје машинског учења: да рачунар **без претходног експлицитног програмирања** може на основу одређеног броја примера **научити како да реши до сада невиђени проблем**. Неуронске мреже су најкоришћенији метод у машинском учењу који је такође коришћен и у овом раду.

Идеја рада била је истренирати неуронску мрежу да генерише текстове у стилу једног од највећих српских аутора, **Иве Андрића**. Желео сам да проверим, да ли би било могуће и у којој мери направити веродостојан модел писања Иве Андрића.

Овим радом покривени су **теоријски темељи** неуронских мрежа који су потребни за разумевање овог рада, **начин имплементације и тренинга**, као и неки од **резултата** који су добијени.

Глава 2

Теоријски темељи

2.1 Машинско учење

Машинско учење је област вештачке интелигенције коју Артур Саумел 1959. године дефинише као област која рачунарима омогућава да **уче без претходног експлицитног програмирања.** Ова област се често описује и као препознавање и проналазак правилности у подацима. Истражује алгоритме који уче из података и доносе одлуке везане за њих. Најчешће се користи у проблемима за које је **врло тешко или немогуће направити експлицитни алгоритам** са добрим перформансама и поузданошћу. Неки од примера за овакве проблеме су: филтрирање нежељених порука у е-пошти, препознавање слова и цифара са слике, препознавање објеката или људи на сликама, вожња аутомобила и многи други.

Слика 2.1: Примери примене машинског учења: *MNIST* (лево, препознавање цифара), *CIFAR-10* (десно, класификација малих слика у 10 категорија)

Проблеми који се изчавају у машинском учењу се могу поделити у три категорије:

- **Учење под надзором** (енгл. *Supervised learning*): Циљ је да рачунар, на основу одређеног броја улазних примера и жељених резултата-излаза, научи генерално правило које ће касније примењивати на нове, до тада невиђене, улазе.
- **Учење без надзора** (енгл. *Unsupervised learning*): Циљ је да рачунар у задатим подацима који нису организовани и означени пронађе правилности.
- **Учење са појачавањем** (енгл. *Reinforcment learning*): Рачунар је у сталној интеракцији са динамичким окружењем у којем мора извршити одређени задатак (као на пример играње игре против противника или аутоматска вожња аутомобила) и након сваке одлуке добија повратну информацију: награду или казну у зависности од одлуке коју је донео.

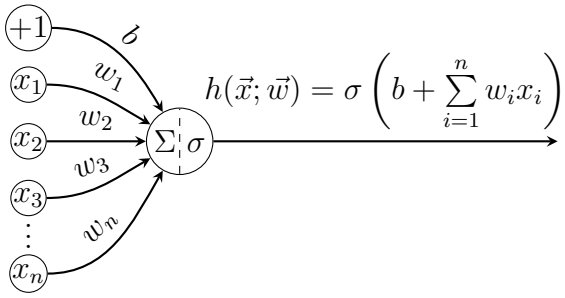
2.2 Неуронске мреже

2.2.1 Дефиниција неуронске мреже

Неуронске мреже су структуре састављене од **више вештачких неурона који су међусобно повезани**. Ове биолошки инспирисане структуре покушавају да имитирају рад неурона у људском мозгу одакле и потиче њихов назив.

2.2.2 Неурон

Сваки неурон рачуна **скаларни производ** (линеарну комбинацију) улазног вектора $\vec{x}$ и његовог тежинског вектора $\vec{w}$, након чега на то додаје **слободан члан** b. На добијени резултат неурон примењује **активациону функцију** чиме се добија излаз h.



Слика 2.2: Неурон

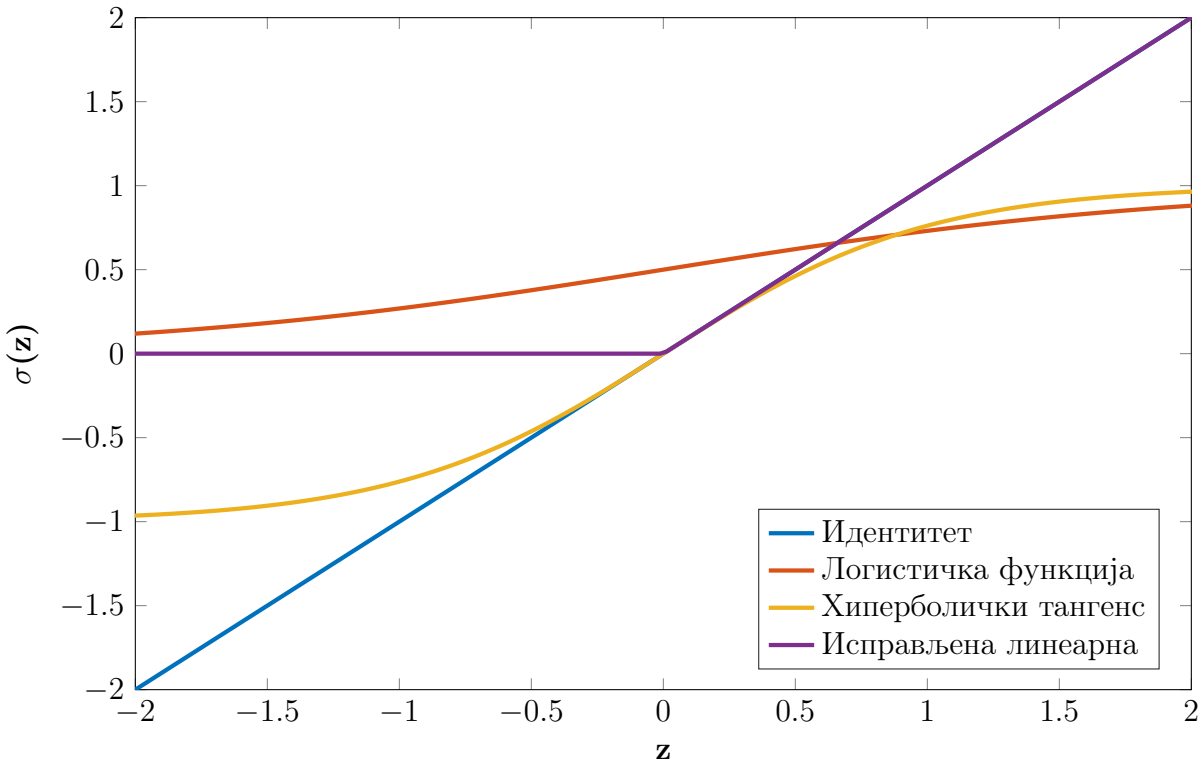
2.2.3 Активационе функције

Активациона функција се бира у зависности од врсте проблема и улазних параметара. Неке од најпопуларнијих активационих функција које се користе су:

- **Индентитет:** $\sigma(z) = z$
- **Сигмоидне функције:**

  логистичка функција: $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

  хиперболички тангенс: $\sigma(z) = \tanh(z)$
- **Исправљена линеарна функција:** $\sigma(z) = \max(0, z)$



Слика 2.3: Активационе функције

На почетку коришћења неуронских мрежа (1950. година), нису се користиле активационе функције, тј. користила се функција идентитета. Међутим, пошто су проблеми који се решавају неуронским мрежама најчешће **нелинеарни**, почињу да се користе сигмоидне функције (назив добијају по карактеристичном облику латиничног слова с).

Вредности хиберболичког тангенса су у интервалу $[-1, 1]$, а логистичке функције у интервалу $[0, 1]$.
Функција која се најћешће користи последњих година је исправљена линеарна функција. Коришћење ове функције је започето као **инжењески начин за представљање нелинеарности** проблема, међутим, ова функција се у пракси показала знатно боље од претходно коришћених функција. Такође, ова функција најбоље осликава рад неурона у људском мозгу: импулс се преноси тек када његова јачина пређе одређену границу.

2.2.4 Од неурона, до неуронске мреже

Међусобним повезивањем неурона настаје неуронска мрежа. Групе неурона се могу поделити у више слојева: **улазни** слој, **скривени** слој (или слојеви) и **излазни** слој. У зависности од сложености проблема се бира број скривених слојева. Уколико је број скривених слојева **већи од 1**, неуронска мрежа се назива **дубоком**.
Повезивање се може извршити на два начина:

- **Ациклично** (енгл. *feedforward*) - Граф повезаности неурона **нема циклуса**. Најчешће су организоване у слојевима, тако да сваки неурон прослеђује резултат ка неуронима из следећег слоја док не стигне до излаза. Сваки слој врши **независну обраду информација и подаци које обрађује морају бити фиксне дужине**. Управо зато се овакве мреже не могу користити за проблеме у којима одговор на тренутно питање зависи од претходних одговора или је улаз произвољне дужине (превод текста или препознавање говора на пример). Примери примене оваквих мрежа су: препознавање цифара или слова са слике, класификација слика и података.

- **Рекурентно** - У графу повезаности **могу постојати циклуси** (ово је начин повезивања који ће бити коришћен у овом раду). Овакво повезивање омогућава обрађивање информација **произвољне дужине**. Један улаз може пролазити кроз више неурона, а потом поново постати нови улаз истог. На тај начин ће **ранији резултати имати утицај на нове**. Примери су: препознавање говора, моделовање језика, превод текста (у примерима језика, за сваки појединачни карактер морамо посматрати и претходне карактере како бисмо извукли неки закључак).

2.2.5 Тренинг неуронске мреже

У наредним деловима овог рада, све информације ће се односити на учење под надзором.
Кључни део приликом израде неуронске мреже пре њене примене јесте тренинг, тј.

проналазак тежинског вектора $\vec{w}$ и слободних чланова b за сваки неурон. Најчешће се ове вредности иницијално случајно генеришу или поставе на одређене познате вредности за које се зна да су погодне како би тренинг био ефикаснији. Алгоритам тренинга тражи најоптималније вредности тежинског вектора и слободних чланова како би **максимизовао моћ предвиђања** мреже.

Функција губитка $\mathcal{L}$ јесте мера колико је тренутно решење (кофигурација $\vec{w}$ и b) удаљено од отималног. Након правилног избора ове функције, потребно је **минимизовати њену вредност**. Приликом овог процеса, користе се већ познати скупови улаза и жељених излаза (тренинг скуп), уз прмену метода алгоритма пропагације уназад и спуста низ градијент.

2.2.6 Алгоритам пропагације уназад

Примена алгоритма градијентног спуста на тренирање неуронских мрежа је **итеративно ажурирање тежина** користећи следећу формулу

$$\vec{w}_{t+1} \leftarrow \vec{w}_t - \eta \nabla \mathcal{L}(\vec{w}) \tag{2.1}$$

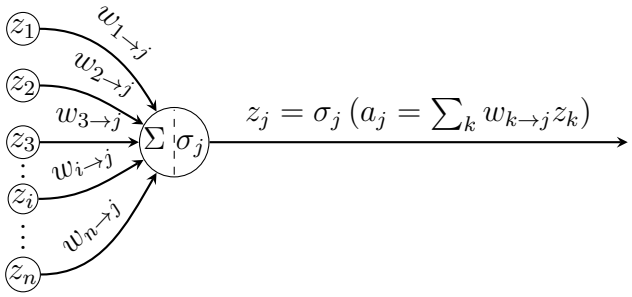
где је $\nabla \mathcal{L}(\vec{w})$ градијент *функције губитка*, t број итерације, а η стопа учења која се фиксира.
Ради поједностављења рачуна, може се сматрати да неурони **немају слободни члан** b. Неопходно је одредити све вредности парцијалних извода

$$\frac{\partial \mathcal{L}(\vec{w})}{\partial w_{i \to j}} \tag{2.2}$$

(за тежину која спаја излаз неурона i са једним од улаза неурона j). Ако ово знамо да урадимо за један познати тренинг пример $(\vec{x}', y')$, ово можемо урадити и за цео тренинг скуп, као суму свих парцијалних извода по свим примерима одвојено.

Први корак (пропагација унапред) подразумева **убацивање овог примера као улаза** за мрежу и израчунавање $y = h(\vec{w}; \vec{x}')$, уз то да за сваки неурон j чувамо активацију a_j и коначан излаз z_j (резултат примене активационе функције овог неурона, σ_j, на активацију a_j, $z_j = \sigma_j(a_j)$):

$$a_j = \sum_k w_{k \to j} z_k z_j = \sigma_j(a_j) \tag{2.3}$$



Слика 2.4: Приказ улаза, тежинских коефицијената, активација и излаза

На основу правила о изводу сложене функције важи:

$$\frac{\partial \mathcal{L}(\vec{w})}{\partial w_{i\to j}} = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_j} \frac{\partial a_j}{\partial w_{i\to j}} \tag{2.4}$$

Други део овог производа је тривијално одредити:

$$\frac{\partial a_j}{\partial w_{i\to j}} = \frac{\partial}{\partial w_{i\to j}} \left(\sum_k w_{k\to j} z_k \right) = z_i \tag{2.5}$$

Даље, проналаском прве стране производа (који ћемо означити са δ_j) за свако j решавамо проблем:

$$\delta_j = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_j} \tag{2.6}$$

$$\frac{\partial \mathcal{L}(\vec{w})}{\partial w_{i\to j}} = z_i \delta_j \tag{2.7}$$

Израчунавањем **од излазних чворова уназад** (пропагација уназад), ове вредности се могу одредити.

Прво разматрамо случај δ_j за **излазне неуроне** j чији је излаз z_j. Поново, применом правила о изводу сложене функције важи:

$$\delta_j = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_j} = \frac{\partial \mathcal{L}(\vec{w})}{\partial z_j} \frac{\partial z_j}{\partial a_j} \tag{2.8}$$

Пошто z_j зависи само од a_j ($z_j = \sigma_j(a_j)$), следи да је $\frac{\partial z_j}{\partial a_j} = \frac{dz_j}{da_j}$, тј. обични извод активационе функције овог неурона у a_j. Дакле: $\frac{\partial z_j}{\partial a_j} = \frac{d\sigma_j}{dx}(a_j) = \sigma_j'(a_j)$.
Са друге стране, пошто функција губитка најчешће третира губитке као збир губитака по сваком неурону посебно, $\frac{\partial \mathcal{L}(\vec{w})}{\partial z_j}$ је директан извод функције губитка за неурон j (остали

сабирци су 0). Он се може директно израчунати, поређењем z_j са очекиваним излазом y' (или делом излаза ако постоји више излазних неурона).
Ове вредности су лаке за израчунавање уколико бирамо активационе функције и функције губитака чији се извод по некој променљивој релативно лако рачуна.

За све **остале неуроне** j (који нису излазни), промена акцтивације a_j директно мења све неуроне k за које постоји тежина $w_{j\to k}$.
Пошто вредности δ рачунамо уназад од излазних неурона и мрежа нема циклуса, можемо вредност δ_j изазити преко већ израчунатих вредности δ_k користећи, поново, правило извода сложене функције:

$$\delta_j = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_j} = \sum_k \frac{\partial \mathcal{L}(\vec{w})}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j} \tag{2.9}$$

$$\frac{\partial a_k}{\partial a_j} = \frac{\partial}{\partial a_j}\left(\sum_i w_{i\to k}\sigma_i(a_i)\right) = w_{j\to k}\frac{d\sigma_j}{dx}(a_j) = w_{j\to k}\sigma_j'(a_j) \tag{2.10}$$

$$\delta_j = \sigma_j'(a_j)\sum_k w_{j\to k}\delta_k \tag{2.11}$$

Алгоритам пропагације уназад приказан укратко:

1. **Пропагација унапред** - убацити $\vec{x}'$ као улаз за мрежу и одредити све активације $a_j = \sum_i w_{i\to j}z_i$ и изразе $z_j = \sigma_j(a_j)$ за све неуроне j у мрежи

2. **Пропагација уназад**

   Излазни неурони j

   $$\delta_j = \sigma_j'(a_j)\frac{\partial \mathcal{L}(\vec{w})}{\partial z_j} \tag{2.12}$$

   Остали неурони j

   $$\delta_j = \sigma_j'(a_j)\sum_k w_{j\to k}\delta_k \tag{2.13}$$

3. **Ажурирање тежинских кеофицијената** - коришћењем алгоритма спуста низ градијент

   $$w_{i\to j} \leftarrow w_{i\to j} - \eta z_i \delta_j \tag{2.14}$$

Изводи најпознатијих активационих функција:

- **Идентитет**

  $$\sigma(z) = z \tag{2.15}$$

$$\sigma'(z) = 1 \tag{2.16}$$

- **Сигмоидне функције**

  логистичка функција

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.17}$$

$$\sigma'(z) = \frac{1}{1 + e^{-z}} \frac{-e^{-z}}{1 + e^{-z}} = \sigma(z)(1 - \sigma(z)) \tag{2.18}$$

  хиперболички тангенс

$$\sigma(z) = \tanh(z) \tag{2.19}$$

$$\sigma'(z) = 1 - \tanh^2(z) = 1 - \sigma(z)^2 \tag{2.20}$$

- **Исправљена линеарна функција:**

$$\sigma(z) = \max(0, z) \tag{2.21}$$

$$\sigma'(z) = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases} \tag{2.22}$$

2.2.7 *One-hot encoding*

One-hot encoding је облик представљања података који је погодан за улаз и излаз у неуронској мрежи која решава проблеме класификације. За стварање овог низа потребно је знати укупан број класа који постоји (или се посматра). *One-hot encoding* једног податка је **низ нула и једне јединице** при чему је његова дужина једнака укупном броју класа, а вредност низа на позицији i је **1 уколико елемент припада тој класи, а 0 у супротном случају**.

Примери:

1. Уколико укупно постоје 4 класе превозних средстава: аутомобил, авион, воз и камион (овим редом), *one-hot encoding* авиона би био $\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$.

2. Дефинишимо скуп класа (вокабулар) којима ћемо редом доделити бројеве:

$$\Big\{(\text{а}, 0),\ (\text{б}, 1),\ (\text{в}, 2),\ (\text{г}, 3),\ (\text{д}, 4),\ (\text{ђ}, 5),\ (\text{е}, 6),\ (\text{ж}, 7)\Big\}$$

На тај начин се текст „жаба” представља $\begin{bmatrix}7 & 0 & 1 & 0\end{bmatrix}$.
Уколико се уради *one-hot encoding*, добијају се 4 низа (по један за свако слово речи).

$$\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

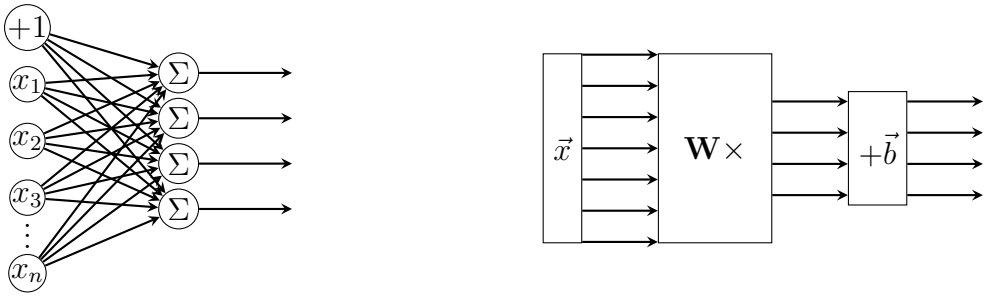
$$\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

Управо на начин приказан у примеру 2 ће бити представљани тренинг примери (улаз и излаз) који се користе за тренирање мреже у овом раду.

2.2.8 Проблеми класификације

Најчешћи проблеми који се решавају неуронским мрежама јесу проблеми класификације. Они су описани **дискретним скупом излаза** тј. класа и циљ је одредити којој класи припада дати улаз.
Посматрајмо најједноставнији класификатор са 4 могуће класе. Сваки од неурона рачуна линерарну комбинацију улаза и тежинских коефицијената што у ствари представља множење вектора улаза са матрицом тежина. На крају се на ово све додаје и слободан члан.



Слика 2.5: Једноставан четворокласни класификатор приказан на два начина

Најчешће се мрежа тренира тако да излаз који мрежа сматра **највероватнијим** има највећу вредност.

$$C = \underset{j}{\operatorname{argmax}} \left\{ b_j + \sum_{i=1}^{n} w_{ij} x_i \right\} = \underset{j}{\operatorname{argmax}} \left(\mathbf{W}\vec{x} + \vec{b} \right)_j \tag{2.23}$$

Проблем који се овде јавља јесте што ове излазне вредности могу узимати било које реалне вредности. На пример, уколико би излаз требало да буде 3. класа, излазни вектор би у идеалном случају био $\vec{y} = \begin{bmatrix} -\infty & -\infty & +\infty & -\infty \end{bmatrix}$ што је врло незгодно представљати и даље користити. Управо из тог разлога се користи трансофрмација излаза **монотоном** функцијом ***softmax***.
Softmax је генерализација раније поменуте логистичке функције.

$$softmax(\vec{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.24}$$

Значајно је приметити да је кодомен ове фукнције $[0, 1]$ и да је $\sum_i softmax(\vec{z})_i = 1$ па се излазне вредности на које је примењена ова функција могу **интерпретирати као вероватноће да је улаз $\vec{x}$ класе i.**

$$y'_i = softmax(h(\vec{x}))_i = \mathbb{P}(\vec{x} \text{ је класе } i) \tag{2.25}$$

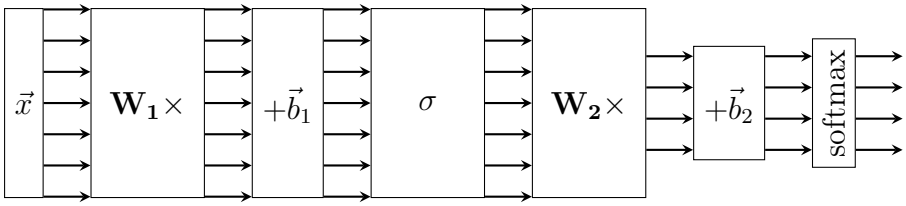
У претходном примеру би сада идеални излаз за трећу класу изгледао овако: $\vec{y} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$.

Приликом коришћења *softmax* је **најпогодније користити и *cross-entropy* функцију фубитка**:

$$\mathcal{L}(\vec{y}, \vec{y'}) = \sum_{i=1}^{K} y_i \log y'_i \tag{2.26}$$

где је K број класа, y жељени излаз током тренинга, а y' излаз на који је примењена *softmax* функција.

Даље додавање нових слојева и стварање дубоке мреже сада је једноставно.



Слика 2.6: Додавање слојева у мрежу

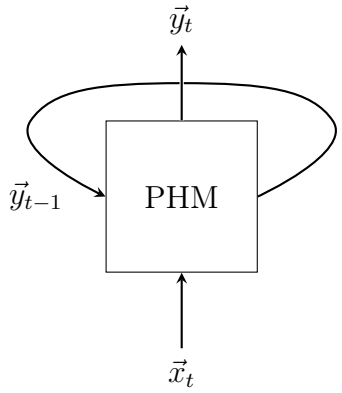
$$C = \underset{j}{\operatorname{argmax}} \left\{ softmax\left(\mathbf{W_2}\sigma\left(\mathbf{W_1}\vec{x} + \vec{b}_1\right) + \vec{b}_2\right)_j \right\} \tag{2.27}$$

2.3 Рекурентне неуронске мреже (РНМ)

2.3.1 Увод у рекурентне неуронске мреже

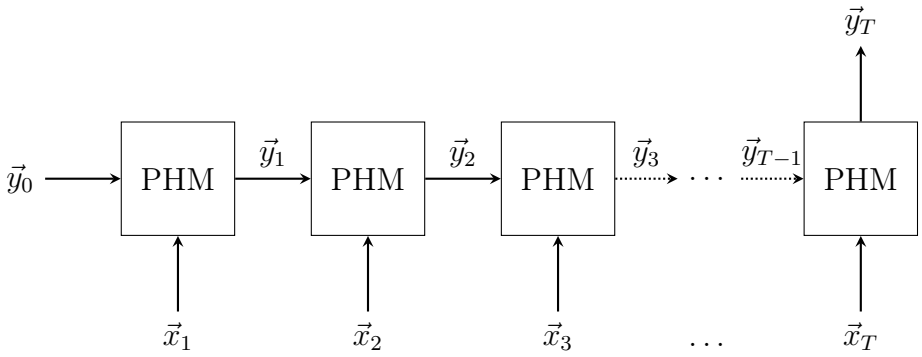
Неуронске мреже о којима је до сада било речи су биле ацикличне мреже чији неурони независно израчунавају излазе.

Приликом читања текста, човек не закључује значење прочитаног само из последњег слова или последње речи, већ **из одређеног броја речи које су претходиле тренутној**. Број речи који је потребан за закључивање контекста може бити произвољан (зависи од реченице до реченице). Ацикличне мреже немају способност оваквог закључивања. Са друге стране, рекурентне неуронске мреже пружају решење управо за овај проблем. Пре свега, рекурентне неуронске мреже дозвољавају обраду ствари које су **произвољне дужине** (истом мрежом). Такође, како један улаз може пролазити кроз више неурона, а потом поново постати нови улаз истог, неурони у рекурентним неуронским мрежама, приликом доношења одлуке, у обзир узимају и **одлуке донете раније**. РНМ налазе примену у проблемима као што су: моделовање језика, генерисање текста, превод, описивање слика.



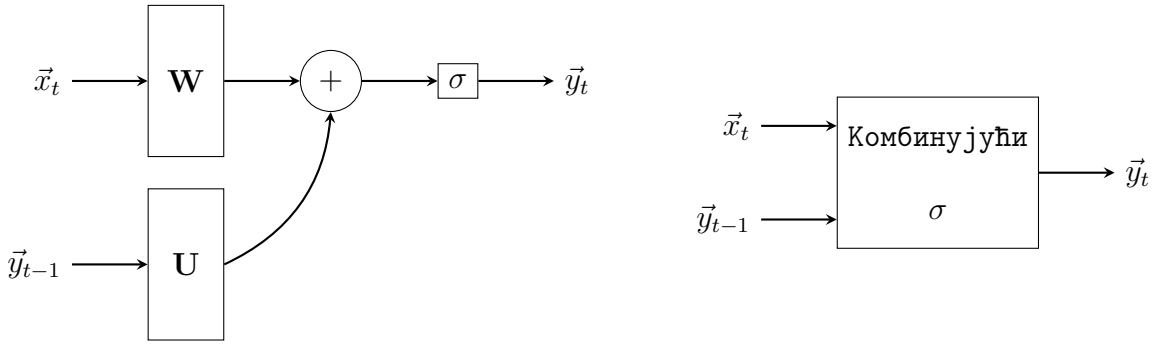
Слика 2.7: Ћелија рекурентне неуронске мреже

Рекурентна неуронска мрежа се може замислити као више еквивалентних РНМ блокова који су повезани (**развој РНМ кроз време**, сваки блок се одвија у једном временском периоду). Сваки излаз $\vec{y}_t$ се рекурентно рачуна користећи $\vec{x}_t$ и $\vec{y}_{t-1}$.



Слика 2.8: Развој рекурентне неуронске мреже кроз време

Свака од ових еквивалентних ћелија (блокова) се понаша као **две потпуно повезане ацикличне неуронске мреже** (без активационих функција, са узалима $\vec{x}_t$ и $\vec{y}_{t-1}$) чији резултати се сабирају и на њих примењује активациона функција како би се добио излаз $\vec{y}_t$. Од сада ћемо ово називати ***„комбинујући блок"***.



Слика 2.9: *Комбинујући блок* приказан на два начина

$$\vec{y}_t = \sigma\left(\mathbf{W}\vec{x}_t + \mathbf{U}\vec{y}_{t-1} + \vec{b}\right) \tag{2.28}$$

За активациону функцију није погодно узимати идентитет јер су проблеми који се решавају нелинеарни. Погоднije функције које се користе јесу **исправљна линеарна и сигмоидне фунције**. Међутим, приликом коришћења ових функција у РНМ долази до проблема као што су **експлодирајући и нестајући градијенти** који ће касније бити детаљније обрађени.

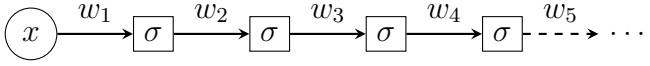
2.3.2 Алгоритам пропагације уназад кроз време

Алгоритам пропагације уназад објашњен у претходном поглављу (за ацикличне неурон-

ске мреже) се не може примењивати на рекурентне неуронске мреже јер у њима постоје циклуси. Међутим, када РНМ развијемо кроз време као што је то приказано у претходној секцији, овај алгоритам се може **рекурентно од краја примењивати** на еквивалентне РНМ блокове што је врло слично примени на класичној ацикличној мрежи.

2.3.3 Проблеми РНМ

Посматрајмо веома једноставну неуронску мрежу где сваки слој има по један неурон који користи исту активацону функцију σ.



Ово јесте веома једноставан пример, међутим, оно што закључимо на њему се **може применити и на остале мреже** јер се оне могу декомпоновати на овакве једноставније мреже. Код РНМ, величина оваквих једноставих мрежа је у најмању руку једнака величини улаза.

Посматрајмо неурон i. Нека је његова активација a_i, а z_i излаз.

$$z_0 = x, \quad a_i = w_i z_{i-1}, \quad z_i = \sigma(a_i) \tag{2.29}$$

Парцијални извод функције фубитка по w_2 изгледао би овако (коришћењем правила извода сложене функције):

$$\frac{\partial \mathcal{L}(\vec{w})}{\partial w_2} = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_2}\frac{\partial a_2}{\partial w_2} = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_2}\frac{\partial (w_2 z_1)}{\partial w_2} = \frac{\partial \mathcal{L}(\vec{w})}{\partial a_2} z_1 \tag{2.30}$$

$$= \sigma(a_1)\frac{\partial \mathcal{L}(\vec{w})}{\partial a_3}\frac{\partial a_3}{\partial z_2}\frac{\partial z_2}{\partial a_2} = \sigma(a_1)\frac{\partial \mathcal{L}(\vec{w})}{\partial a_3}\frac{\partial (w_3 z_2)}{\partial z_2}\frac{\partial \sigma(a_2)}{\partial a_2} \tag{2.31}$$

$$= \sigma(a_1)\sigma'(a_2)w_3\frac{\partial \mathcal{L}(\vec{w})}{\partial a_4}\frac{\partial a_4}{\partial z_3}\frac{\partial z_3}{\partial a_3} = \sigma(a_1)\sigma'(a_2)w_3\sigma'(a_3)w_4\frac{\partial \mathcal{L}(\vec{w})}{\partial a_5}\frac{\partial a_5}{\partial z_4}\frac{\partial z_4}{\partial a_4} \tag{2.32}$$

$$= \sigma(a_1)\sigma'(a_2)w_3\sigma'(a_3)w_4\sigma'(a_4)\ldots \tag{2.33}$$

1. **Нестајући градијент**
   За логистичку функцију важи: $\sigma'(x) = \sigma(x)(1-\sigma(x)) \implies |\sigma'(x)| \leq 0.25$.
   За хиперболички тангенс важи: $\sigma'(x) = 1 - \sigma(x)^2 \implies |\sigma'(x)| \leq 1$.
   Када погледамо израз за парцијални извод функције фубитка по w_2, можемо уочити да се множењем великог броја извода активационих функција, од којих је **сваки мањи од 1, брзо достиже 0** и самим тим ***градијент нестаје***.

2. **Експлодирајући градијент**
   Извод исправљене линеарне функције је 0 или 1, али први члан $\sigma(a_1)$ **може расти без ограничења**. Ово није проблем код ацикличних мрежа, међутим, код РНМ које деле тежине ово представља велики проблем. Сумирањем великог броја вредности које нису ограничене долази до ***експлодирања градијента***.

   Проблем експлодирајућег градијента се решава техником која се зове *gradient clipping* којом се градијент нормира одређеном границом.

2.3.4 *LSTM* мреже

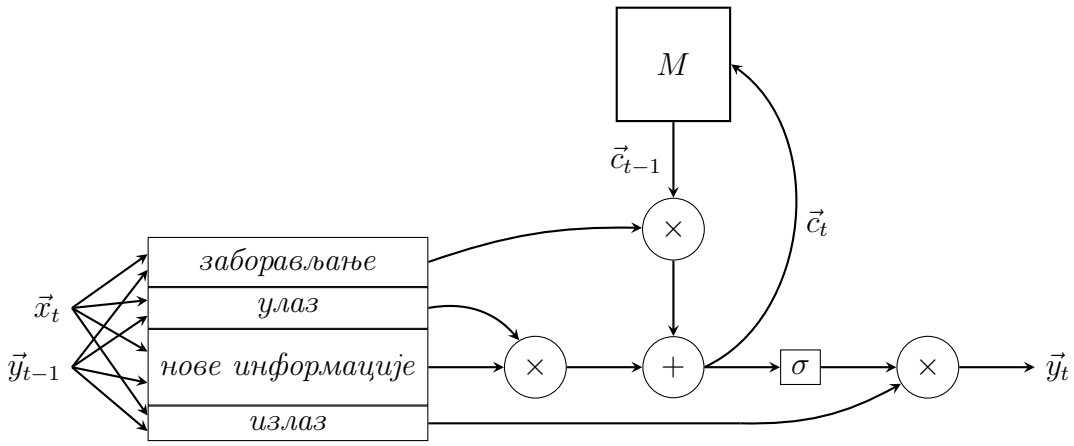
Главна идеја РНМ јесте да омогући **произвољан број улазних података** и повеже **раније информације** са тренутним задатком. У зависности од ситуације до ситуације, РНМ могу ово повезивање извршити са различитом успешношћу.

Посматрајмо следећи пример.
Задатак мреже јесте да претпостави која реч ће следити задатом тексту. У реченици „На небу је *облак*", реч *облак* се може претпоставити без неког ширег контекста. У реченици „У *Србији* сам живео и радио 3 године због чега сам научио *српски*". За претпостављање речи *српски*, потребан је много шири контекст, а самим тим ће **градијент постати практично 0** док се рекурентно рачуна до речи *Србији*.

Што је већи приказани размак (контекст) између релевантних информација, то је за РНМ теже да то и науче.

Међутим, специјална врста РНМ, *LSTM* мрежа (у буквалном предводу са енглеског значи *мрежа са дугом краткорочном меморијом*), **нема овај проблем због начина на који је имплементирана**. Управо из тог разлога је она данас најкоришћенија врста рекурентне неуронске мреже.

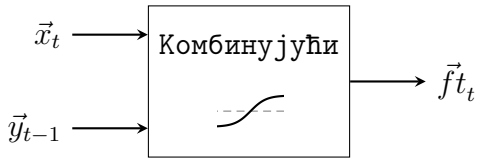
За разлику од класичних РНМ, у *LSTM* мрежама се уводи још једна ћелија, **ћелија меморије** која чува информације између корака. Такође се рачуна и колика количина **нових информација** ће ући у меморијску ћелију, колико ће бити **избачено** из ње, а и колика количина ће **изаћи** из *LSTM* модела. Ово је омогућено коришћењем **капија**. На овај начин мрежа учи да заборавља ствари када је то потребно (решење нестајућег градијента).



Слика 2.10: Приказ *LSTM* мреже

2.3.5 Капије *LSTM* мрежа

Блок нових информација ($\vec{ft}_t$) одређује ново стање ћелије које ће заменити старо у одређеном степену (у зависности од осталих капија). То је *комбинујући блок* који на основу $\vec{x}_t$ и $\vec{y}_{t-1}$ (као и тежина W и U) даје излаз у интервалу $[-1, 1]$ (активациона функција је хиперболички тангенс).



$$\vec{ft}_t = \tanh\left(\mathbf{W}_{ft}\vec{x}_t + \mathbf{U}_{ft}\vec{y}_{t-1} + \vec{b}_{ft}\right) \tag{2.34}$$

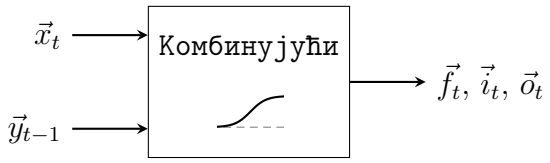
Капија заборављања ($\vec{f}_t$) одређује степен у ком ће досадашње информације бити **одабачене**.

Капија улаза ($\vec{i}_t$) одређује степен у ком ће нове информације бити **узете у обзир**.

Капија излаза ($\vec{o}_t$) одређује степен у ком ће информације изаћи из *LSTM* мреже.

Капије заборављања, улаза и излаза су *комбинујући блокови* који на основу $\vec{x}_t$ и $\vec{y}_{t-1}$ (као и тежина W и U) дају излаз у интервалу $[0, 1]$ (активациона функција је логистичка

функција), где би **1 представљало да информације буду у потпуности сачуване/узете у обзир, а 0 да оне буду у потпуности заборављене/одбачене**.



$$\vec{f_t} = logistic\left(\mathbf{W}_f\vec{x}_t + \mathbf{U}_f\vec{y}_{t-1} + \vec{b}_f\right) \tag{2.35}$$

$$\vec{i_t} = logistic\left(\mathbf{W}_i\vec{x}_t + \mathbf{U}_i\vec{y}_{t-1} + \vec{b}_i\right) \tag{2.36}$$

$$\vec{o_t} = logistic\left(\mathbf{W}_o\vec{x}_t + \mathbf{U}_o\vec{y}_{t-1} + \vec{b}_o\right) \tag{2.37}$$

Финално, врши се **промена стања ћелије**, а на основу новог стана ћелије се **одређује и излаз**.

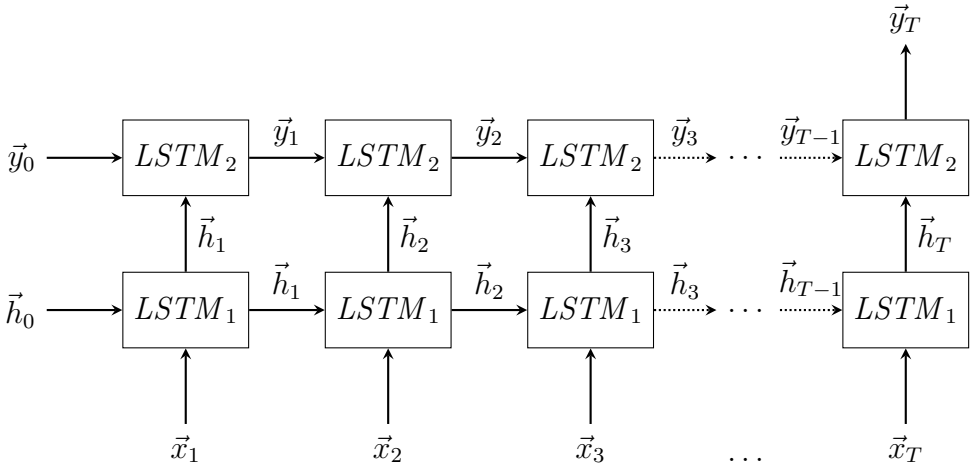
$$\vec{c_t} = \vec{ft}_t \otimes \vec{i_t} + \vec{c}_{t-1} \otimes \vec{f_t} \tag{2.38}$$

$$\vec{y_t} = \tanh\left(\vec{c_t}\right) \otimes \vec{o_t} \tag{2.39}$$

** $\otimes$ је element-wise производ матрица (вектора), тј. за $M_1 \otimes M_2 = M$ је $M[i][j] = M_1[i][j] * M_2[i][j]$*

2.3.6 Дубоке *LSTM* мреже

Дубока неуронска мрежа јесте мрежа која има више од једног скривеног слоја. Додавање слојева у *LSTM* мрежу се врши повезивањем међуизлаза првог слоја са улазима следећег.



Слика 2.11: Вишеслојна $LSTM$ мрежа

Глава 3

Имплементација и тренинг

3.1 Коришћена библиотека

За имплементацију је коришћена **библиотека *Keras***. Дизајнирана је како би програмерима омогућила **брзо експериментисање са дубоким неуронским мрежама** које су генерално тешке за имплементацију од нуле. *Keras* користи програмски језик *Python*, а за извршавање се ослања на две познате библиотеке за машинско учење - *TensorFlow* и *Theano*.

3.2 Идеја, имплементација и тренинг

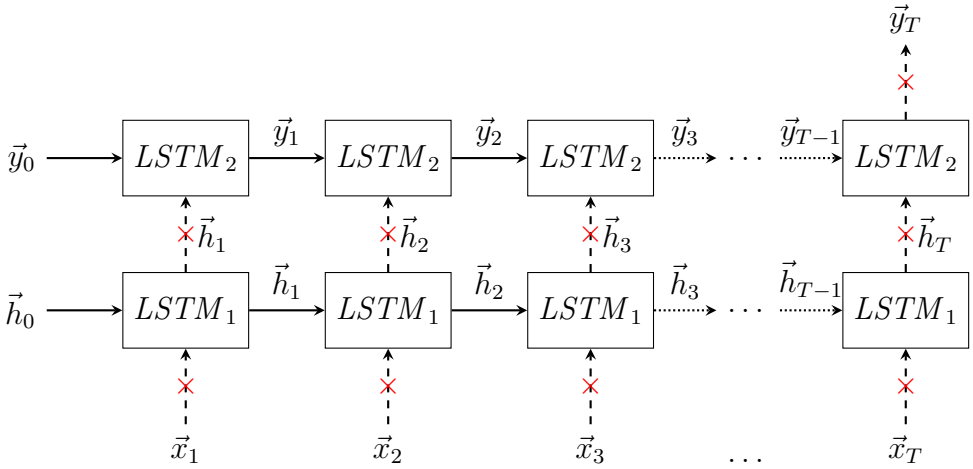
Идеја рада била је истренирати неуронску мрежу да **генерише текстове у стилу једног од највећих српских аутора, Иве Андрића**. У ту сврху, коришћена је ***LSTM* рекурентна неуронска мрежа**.

За тренинг мреже коришћено је Андрићево дело **„На Дрини ћуприја"** које је дужине 657042 карактера. На основу текста је одређен **вокабулар** који се састојао од **76 јединствених карактера** (мала и велика латинична слова, размак, знакови интерпункције и још неколико других карактера). Сваком карактеру речника је додељен јединствени број од 0 до 75. Свака секвенца текста фужине 50 карактера је прво представљена помоћу бројева од 0 до 75, а затим је сваки карактер представљен *one-hot encoding*-ом. Низ излазних примера је низ *one-hot encoding* карактера који следе одговарајућим секвенцама из улаза.

Мрежа је имала **улазни слој, 2 *LSTM* слоја и излазни слој**. За излазни слој, коришћена је ***softmax* функција**. За функцију губитка је коришћена категорична

***crossentropy* функција**. Тренинг је извршен у **100 епоха** (100 пролазака кроз тренинг примере).

Ради ефикаснијег тренинга, коришћене су технике **случајног одбацивања појединих веза (*dropout*)** и ***Adam optimizer*** *(Kingma, Ba (2014))*.
Случајно одбацивања појединих веза (*dropout*) форсира мрежу да се не ослања на постојање одређених неурона. Сигурно је одбацивање само нерекурентних веза.



Слика 3.1: Случајно одбацивања појединих веза (*dropout*)

Генерисање текста врши се тако да за дату почетну секвенцу текста, мрежа **предвиђа карактер који следи**, а затим на основу новодобијене секвенце поново предвиђа нови карактер све док не генерише тражени број карактера.

3.3 *Keras* код архитектуре мреже

```
1  input = Input(shape = (seq_len, vocab_len))
2  h_1 = LSTM(512, return_sequences = True)(input)
3  h_2 = LSTM(512, dropout_W = 0.5)(h_1)
4  out = Dense(vocab_len, activation = 'softmax')(h_2)
5  model = Model(input = inp, output = out)
6  model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

Глава 4

Резултати

4.1 Примери генерисаних текстова по епохама

Епоха 5

Najposle, ta njegova bojazan pretvorila se u pravu saciju, gleda. Zastavljas je poneset okrupljiva, u samo sukutuje ga bilo brza. A u pod tak cilnim svetama iza mnoge sa brizama videlo izvrzene sade udarinu. Na vreme durao je platuvala vlasta candak, hvatao, oni pripriti sve zivoti: vlada oko obehnuzenja ili da izvesti kao mune koje mu je tu posio, tako padna mogao. I ko bao kako vreme on kad ugleda, nista srecno kad. Svakoga oblike i nepovelo.

Епоха 20

Najposle, ta njegova bojazan pretvorila se u pravu i zivlju neznatno titinja. Gledajuci tebe i sporo i rnadjeno prisvojno osudjenike, ali iz svoga sudbina iskrenog stuba koji zaostaju na kapiji. Sad se moze razgovarati nisu bili ni u vlasti. Ali, kako su mu se drzali na mostu i savrseni sudbina potpuno sumnjiva. Izgledalo se kao da ce pored retkih, iako snazni novca, On je jedini poveren okim cutkeg crne zene sve je veci deo cardak; na prvom selu...

Епоха 40

Najposle, ta njegova bojazan pretvorila se u pravu tursku, tup u Sembi — Cigane Hrinja; neka zna? — Ah, kakhkrkha! Samo drugi jasni ga nesto pitaju. I tu iz daleka, preko dve strane same da bi su pristali a u varosi i picu i nadali. Jos po nekoliko pogleda i pevacka drzava, u kojoj sedelji bio je tvrdo gornje u zemlju basi nece ni to sto preci. Bili je i mirnije na sebe i zvonovronosti. Njegove nasike i mrsava i promena znacu po jedan sat. Ali...

Епоха 60

Najposle, ta njegova bojazan pretvorila se u pravu, ovakvu karavanu i oko usku vezinov dogore. Glava bi ga nedoglednim i bezizlazim, on je ostetio sa mesaju sitne razlike, ali u sofi, ali to

ce se na kraju mosta, i pomadja svoje beskrajnjem drvenom balku nerazumljiva, maloduznin, da nisu iz zaptiju. Kad je hodza namarlo prednje da procitao teskim zadahom. Tako je vise popuNila kratko, jer izgledaju tako lakse i oprezno. Sa Suljama i Zakolima do...

Епоха 80

Najposle, ta njegova bojazan pretvorila se u pravu, budu vodeni dvojice dece bije, cesto i gotovcava zastade. Malo se zaista na kapiji sa konjima i da predje ustanjema u okajama snaga da se primeci o mostu, dok ne podnosi vec govorilo ga predeo da ga saikom pred mostom obasjati i neprestano gubi produze brige o Kameniti sut. On je bio prvi prizor Mahaiz, zastoji mrak, da se vila pravo misli d hotela. Zato je proveo pod uticajem cudise, seljaci iz...

Епоха 100

Najposle, ta njegova bojazan pretvorila se u pravu, osecalo te letnje i koljenti. I sam, mogucno padi ovaj most, bez ikakve mesta i prosla konako. Tako je covek izmedju dve ratrice zaista menilo do koletnog pitanja. Plevljak je izdavao polomika, u avlijskom stalu i tako se dvojica osudjenih ljudi, da to znaci da se odmori, zajedno sa muzikom, i kao pop njegovoj ducanje i da je on neprostan konaka. Na nov noc ili dan uzdrzljivost pijanca ga nego...

Дати примери текстова су генерисани коришћењем исте почетне секвенце текста. Примећује се да мрежа већ у првих неколико епоха успева да научи реченичну констукцију, коришћење великог слова, размака, зареза... Из епохе у епоху, све је мања количина непостојаних речи.

У даљем тексту се налазе још 5 примера генерисаних текстова из епохе 100.

Dodje ovako izuzetna godina sa narocito srecnim ipomocnim skelevinom, i obudjenim glasom kad gubi ne samo po jedan dolan nego i lekose gradjevinu i primao je sve prirodjeno kako se to dolazilo od njegove formacije koja je ona dva dana na kapiji postao hodza, ali da vise nije smatrao. Deli stat ma prodje i bez stege i uzivanje, da kaze ne varava i koje pomisliti da cinite stakla na obroncima ali neidore i razgovor onog stoleca u jedan od onih sum...

Kad je stigao do kraja, skocio je na drum i gledao znaju da nista da izbija, pod njegovim glasom gledajuci ga i preko kolaca, u prva stihova koja ce on pretvori ulice, ali da devojka spusta samo ispod njega kao neku pijacu i prividno se za poslednjim shvatovima. Sa svakim danom sede oprasti, da se otmu bolest pred nekom strascu pendose i otvori koja su im svi putnici pocesenili nestalo klone i promicu na konjima. Bez obzarica po dva koja se mlad...

Tako su, kako prica kazuje, postale duboke rijeke i po okolnama odvojenog hauba i svojih poslova, novi ljudi stemiju na kapiji jednu visoku polovine tela streperu drvena. Istinja se desavaju svuda na svom mestu, pricao je s njega sve iznesedjeno stvorenje kao mari zenskom kaslju. Stare ga na koga ce biti od njega je rec bilo lakse dodosao u svojoj lepoti. Tako su se odele nabusili i sve u casoj polozivi koji opusteni na smrt, jos je bar po onu st...

Jedino su granate koje su pogadjale sam kolovoz na mostu, poslednji sinovi, ulazi na kapiji, za vreme stvarione ratove i studenti sa dobrocudnom lice, secanje koma ne samo jos dva vojnika i kapija ne moze da javno. Nikad nije proko branilo na njegove pomocnika: — Ni, smo nastavis, nego da ovde biva sta ce dalje biti odmah iznad svoje postolosti nije mogucno odazno zadovoljeni ovom dugom svetu nemir, nego i jedan mali bregove bije prolecno. Nema...

Sve to dok zrna lete visoko i gadjaju po okolini, ali njegov prifori sa oblikim, cenim onim bregovima, koji su se smatrali pre regruta i stale da treba da donesu njegovu hodzinu gradjana. A on je tesko odavde, onda vojnike i podozice od svog zivota, moze biti da vidi coveka lici na kapiji. Ali vecina je bilo ljudi u glavi deo izdzvojeno ovde sto bi poslio. Prokleti lepe jos sne ruke na prazan pored njega, potpuno sedisnu, izlazi preko mosta, ci...

Глава 5

Закључак

Иако до сада тема којом се нисам бавио, неуронске мреже су ме заинтересовале да дубље проучим и разумем **начин њиховог рада**. У овом раду сам укратко приказао читаву **теоријску основу** потребну за разумевање принципа рада и имплементације неуронске мреже. Такође, **имплементирана је и истренирана** *LSTM* мрежа која релативно успешно (али не и толико смислено) генерише текстове у стилу Иве Андрића.

Могућности за евентуални даљи рад јесу даље тренирање мреже како би се постила већа прецизност, као и промене константи које би утицале на другачији исход.

Желео бих и да се **захвалим мом ментору, Петру Величковићу**, на издвојеном времену и обезбеђивању литеруре, *tikz* графика, објашњења нејасних ствари и најбитније, рачунара са довољним перформансама за извршавање тренинга имплементиране мреже.

Литература

[1] Wikipedia. 2017. *Machine learning.* Доступно на: `https://en.wikipedia.org/wiki/Machine_learning`. [Прегледано 25.05.2017].

[2] Wikipedia. 2017. *Artificial neural network.* Доступно на: `https://en.wikipedia.org/wiki/Artificial_neural_network`. [Прегледано 25.05.2017].

[3] Петар Величковић. 2017. *Deep learning for complete beginners: recognising handwritten digits.* Доступно на: `https://cambridgespark.com/content/tutorials/deep-learning-for-complete-beginners-recognising-handwritten-digits/index.html`. [Прегледано 25.05.2017].

[4] Christopher Olah. 2015. *Understanding LSTM Networks.* Доступно на: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. [Прегледано 25.05.2017].

[5] WILDML, Denny Britz. 2015. *Recurrent Neural Networks Tutorial, Part 1 - Introduction to RNNS.* Доступно на: `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`. [Прегледано 25.05.2017].

[6] Петар Величковић. 2016. *Припремни материјали за Недељу информатике: Неуралне мреже.* Доступно на: `http://www.csnedelja.mg.edu.rs/static/resources/v3.0/pripremni_materijali.pdf`. [Прегледано 25.05.2017].

[7] Петар Величковић. 2017. *A trip down long short-term memory lane.* Доступно на: `https://github.com/PetarV-/a-trip-down-lstm-lane`. [Прегледано 25.05.2017].