|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ** |  |

Гаврило Дрљача

**Употреба вештачке неуронске мреже у семантичкој класификацији твитова**

ЗАВРШНИ РАД

- Основне академске студије -

Нови Сад, 2016

|  |  |
| --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 |
| **КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА** |

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | |  | |
| Тип записа, **ТЗ**: | |  | |
| Врста рада, **ВР**: | |  | |
| Аутор, **АУ**: | |  | |
| Ментор, **МН**: | |  | |
| Наслов рада, **НР**: | |  | |
| Језик публикације, **ЈП**: | |  | |
| Језик извода, **ЈИ**: | |  | |
| Земља публиковања, **ЗП**: | |  | |
| Уже географско подручје, **УГП**: | |  | |
| Година, **ГО**: | |  | |
| Издавач, **ИЗ**: | |  | |
| Место и адреса, **МА**: | |  | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | |  | |
| Научна област, **НО**: | |  | |
| Научна дисциплина, **НД**: | |  | |
| Предметна одредница/Кqучне речи, **ПО**: | |  | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | |  | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | |  | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: |  |
|  | Члан: |  | Потпис ментора |
|  | Члан, ментор: |  |  |

Образац **Q2.НА.04-05** - Издање 1

|  |  |
| --- | --- |
|  | UNIVERSITY OF NOVI SAD ⚫ **FACULTY OF TECHNICAL SCIENCES**  21000 NOVI SAD, Trg Dositeja Obradovića 6 |
| **KEY WORDS DOCUMENTATION** |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | |  | |
| Type of record, **TR**: | |  | |
| Contents code, **CC**: | |  | |
| Author, **AU**: | |  | |
| Mentor, **MN**: | |  | |
| Title, **TI**: | |  | |
| Language of text, **LT**: | |  | |
| Language of abstract, **LA**: | |  | |
| Country of publication, **CP**: | |  | |
| Locality of publication, **LP**: | |  | |
| Publication year, **PY**: | |  | |
| Publisher, **PB**: | |  | |
| Publication place, **PP**: | |  | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | |  | |
| Scientific field, **SF**: | |  | |
| Scientific discipline, **SD**: | |  | |
| Subject/Key words, **S**/**KW**: | |  | |
| **UC** | |  | |
| Holding data, **HD**: | |  | |
| Note, **N**: | |  | |
| Abstract, **AB**: | |  | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: |  |
|  | Member: |  | Menthor's sign |
|  | Member, Mentor: |  |  |

Obrazac **Q2.НА.04-05** - Izdanje 1

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Број: |
|  |
| **ЗАДАТАК ЗА МАСТЕР РАД** | Датум: |
|  |

*(Податке уноси предметни наставник - ментор)*

| СТУДИЈСКИ ПРОГРАМ: |  |
| --- | --- |
| РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: |  | Број индекса: |  |
| Област: |  | | |
| Ментор: |  | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; | | | |

**НАСЛОВ МАСТЕР РАДА:**

|  |
| --- |
|  |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за: - Студента; - Ментора |

|  |  |  |
| --- | --- | --- |
|  | УНИВЕРЗИТЕТ У НОВОМ САДУ ⚫ **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  21000 НОВИ САД, Трг Доситеја Обрадовића 6 | Број: |
|  |
| **ЗАДАТАК ЗА ИЗРАДУ ЗАВРШНОГ (BACHELOR) РАДА** | Датум: |
|  |

*(Податке уноси предметни наставник - ментор)*

| Врста студија: | 1. Основне академске студије 2. Основне струковне студије |
| --- | --- |
| Студијски програм: |  |
| Руководилац студијског програма: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: |  | Број индекса: |  |
| Област: |  | | |
| Ментор: |  | | |
| НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА  ИЗДАЈЕ СЕ ЗАДАТАК ЗА ЗАВРШНИ (Bachelor) РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:   * проблем – тема рада; * начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна; * литература | | | |

**НАСЛОВ ЗАВРШНOГ (BACHELOR) РАДА:**

|  |
| --- |
|  |

**ТЕКСТ ЗАДАТКА:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
| Руководилац студијског програма: | Ментор рада: |
|  |  |

|  |
| --- |
| Примерак за: - Студента; - Ментора |

Садржај

[1. Увод 2](#_Toc461374614)

[2. Теоријске основе 3](#_Toc461374615)

[2.1 Машинско учење и класификација 3](#_Toc461374616)

[2.2 Класификационе технике 4](#_Toc461374617)

[2.2.1 Бајесов класификатор 4](#_Toc461374618)

[2.2.2 Стабла одлучивања 5](#_Toc461374619)

[2.2.3 Support Vector Machines (SMV) 6](#_Toc461374620)

[2.2.4 k-Nearest Neighbor Learning (kNN) 7](#_Toc461374621)

[2.2.5 AIRS – A Resource Limited Immune Classifier 7](#_Toc461374622)

[2.2.6 Neural Networks 7](#_Toc461374623)

[2.3 Вештачке неуронске мреже 8](#_Toc461374624)

[2.3.1 Основни елементи и појмови ВНМ 9](#_Toc461374625)

[2.3.2 Обучавање ВНМ 10](#_Toc461374626)

[2.3.3 Архитектуре, структуре и топологије ВНМ 10](#_Toc461374627)

[2.3.4 Конволуционе вештачке неуронске мреже 13](#_Toc461374628)

[2.4 Обрада природних језика 16](#_Toc461374629)

[2.4.1 Историјат 16](#_Toc461374630)

[2.4.2 Употреба машинског учења 17](#_Toc461374631)

[2.4.3 Примене 18](#_Toc461374632)

[2.5 Твитер 21](#_Toc461374633)

[2.5.1 Опис 21](#_Toc461374634)

[2.5.2 Проблематика класификације твитова 21](#_Toc461374635)

[2.6 Програмски језик Phyton 21](#_Toc461374636)

[2.6.1 Историјат 21](#_Toc461374637)

[2.6.2 Карактеристика и филозофија 21](#_Toc461374638)

[2.6.3 Синтакса и семантика 21](#_Toc461374639)

[2.6.4 Библиотеке (Керас итд.) 21](#_Toc461374640)

[3. Литература 21](#_Toc461374641)

# Увод

У овом раду је извршено формално специфицирање и имплементација система за семантичку класификацију твитова.

Рад је подељен у 5 логичких целина.

У првом поглављу је описана сама структура рада, као и теме које су обрађене у свакој од целина које су предвиђене његовом структуром.

У другом поглављу су представљене теоријске основе чије разумевање је неопходно како би се пратила даља разрада и проблематика рада. ~~Поглавље почиње дефинисањем аналогне слике и процес настајања дигиталне слике, а након тога су формално дефинисане обрада и анализа дигиталне слике. У другом делу поглавља су дате теоријске основе вештачких неуронских мрежа. У последњем делу овог поглавља је описан програмски језик Пролог.~~

У трећем поглављу је дата формална спецификација и имплементација система за семантичку класификацију твитова. ~~У првом делу овог поглавља је описан концептуални модел целог система и представљени сви његови подсистеми. У другом делу поглавља је описан подсистем за обраду дигиталне слике. У трећем делу поглавља је описан подсистем за анализу дигиталне слике. У четвртом делу је описан подсистем за оптичко препознавање и формирање математичког израза. Последњи подсистем је описан у петом делу овог поглавља, односно подсистем за симболичко диференцирање математичког израза. Треће поглавље је закључено описом корисничког интерфејса апликације која је имплементирана у складу са спецификацијом која је представљена у претходним деловима поглавља. Апликација је имплементирана на~~ *~~.NET~~* ~~платформи у језику~~ *~~C~~*~~#.~~

У четвртом поглављу је извршена валидација и верификација рада система. ~~У поглављу су прво представљене две референтне машине на којима ће упоредо бити вршено тестирање. Након тога је извршена валидација рада система, односно тестирање тачности над простим и комплексним математичким конструкцијама. Поглавље се завршава тестом перформанси. Тест перформанси је урађен на две претходно описане тестне машине различитих карактеристика и извршено је поређење добијених резултата.~~

У петом поглављу је извршен осврт на имплементиране функционалности система, дата су закључна разматрања и даљи правци развоја и истраживања.

После петог поглавља је представљен кратак преглед свих коришћених скраћеница које се спомињу у раду, као и кратка биографија аутора.

Рад је закључен навођењем литературе која је коришћена у раду. Претходно је усвојено да се користи *IEEE* стил цитирања референци.

# Теоријске основе

У овом поглављу је дат опис свих теоријских концепата чије разумевање је неопходно како би се пратила проблематика рада, као и принципи функционисања коначног решења. У првом делу је описан појам семантичке класификације са освртом на проблематику твитера и начина писања на њему. У другом делу су дате теоријске основе вештачких неуронских мрежа, са описом концепата из те области који се најчешће користе у семантичкој класификацији. У трећем делу је описан програмски језик „Пролог“, који ће бити коришћен у имплементацији подсистема за диференцирање математичких израза (поглавље 3.5).

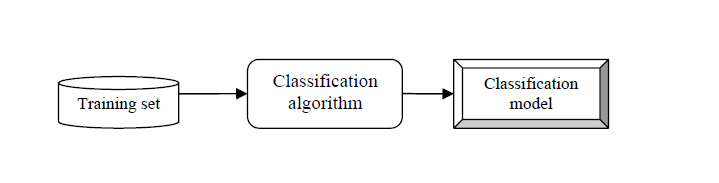
## Машинско учење и класификација

Семантичка класификација представља процес разврставања носилаца семантике (речи, израза, реченица итд. ) у семантичке класе. Семантичка класа садржи речи које деле неко семантичко својство. На пример код именица постоје две подкласе, конкретне и апстрактне. Конкретне обухватају људе, биљке, животиње, материјале и објекте док се апстрактне именице односе на концепте као што су квалитети, акције и процеси. Такође, семантичке класе се могу пресецати, на пример пресек између „жене“ и „младо“ је „девојка“. [1]

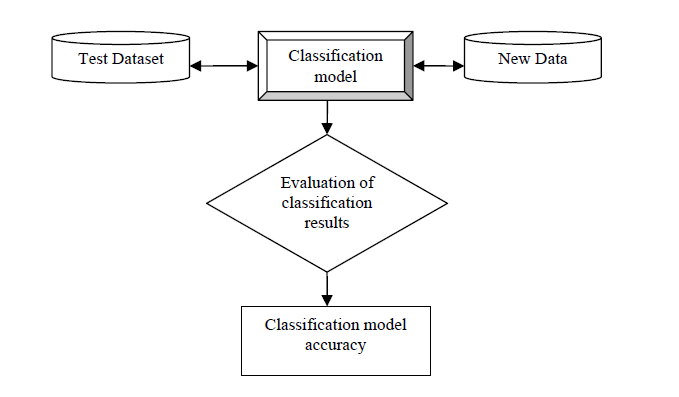
From the invention of computers, a high interest was showed by scientists, in constructing computer programs that can “automatically improve with experience”. Since then, many algorithms have been invented in this field of computer science which is known as *Machine Learning*. Classification is a subfield of machine learning and therefore, the techniques that are used in the latter are a subset of the general techniques of machine learning tasks.

*Classification* can be described as a function that maps (classifies) a data item into one of the several predefined classes [2]. A well-defined set of classes and a training set of pre-classified examples characterize the classification. So, the classification process is a 2-step procedure:

* *Training*. In this step a model is being constructed, describing a predefined set of data classes. The training data are analyzed by a classification algorithm in order the model to be constructed.



* *Classification.* In this step testing data are used in order to calculate the model accuracy. There are several methods to estimate the classifier accuracy. The test data are selected randomly and they are independent from each other. The model classifies the test data and then the known class label is compared with the model prediction about the class. The model accuracy in a specified test dataset is the percentage of the test dataset which has been classified correctly by the learning model.

**

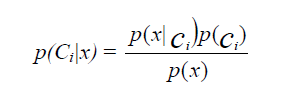
If the accuracy of the model is acceptable, the model can then be used to classify other data items. So, we could say that the goal of the classification is to induce a model that can be used to classify future data items whose classification is unknown [2].

## Класификационе технике

A number of classification techniques have been developed. We now review some of these techniques, such as: Bayesian classifier, Decision Trees, AIRS (Resource Limited Immune Classifier System), k-Nearest Neighbor and Support Vector Machines. All of them are used in the project in conjunction with our semantic similarity metrics.

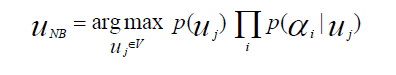
### Бајесов класификатор

Bayesian Classifier is among the most effective known algorithms for text document classification. It is based on Bayesian statistical classification theory. The aim is to classify a sample *x* in one of the known classes C1, C2… Cn, using a probability model defined according to Bayes theory. Each category is characterized by a prior probability of observing the category Ci. Also, we assume that a given sample *x* belongs to a category Ci with the conditional probability density function *p(x|Ci)*∈*[0,1]*. Then, using the above definitions and based on Bayes formula, we define the posterior probability:



An input pattern is classified into a category with the highest posterior probability [3].

The simplest Bayesian Classifier is the *Naïve Bayes Classifier*. “The *Naïve Bayes Classifier* is based on the simplifying assumption that the attribute values are conditionally independent given the target value. The assumption is that given the target value of the instance, the probability of observing the conjunction *α1, α2... αn* is just the product of the probabilities for the individual attributes: *p(α1, α2... αn |uj ) =* Π*p(αi1|uj )* ” [3]. So, the target value output by Naïve Bayes Classifier (uNB) is:

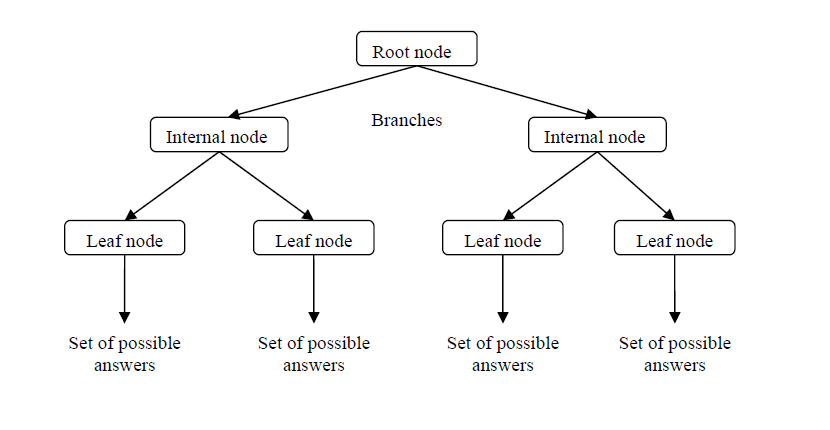


Theoretically, Bayesian Classifiers have the lowest error percentage in comparison with all classifiers. In practice, however, this is not always true, because of the conditional independence assumption which has been made and the lack of available data for the accurate calculation of the conditional probabilities. Nevertheless, researches have shown that Naïve Bayes Classifier is competitive with other well known classifiers, such as Decision Trees and Neural Networks [4].

### Стабла одлучивања

*Decision Trees* are one of the widely used techniques for classification and prediction. A number of popular classifiers construct decision trees to generate classification models.

A decision tree is constructed based on a training set of pre-classified data. Each internal node of the decision tree specifies a test of an attribute of the instance and each branch descending of that node corresponds to one of the possible values for this attribute. Also, each leaf corresponds to one of the defined classes. The procedure to classify a new instance using a decision tree is as follows: starting at the root of the tree and testing the attribute specified by this node, successive internal nodes are visited until a leaf is reached. At each internal node, the test of the node is applied to the instance. The outcome of this test at an internal node determines the branch traversed and the next node visited. The class for the instance is the class of the final leaf node [4].



Several algorithms for constructing decision trees have been developed. Some of the most widely known algorithms are: ID3 [3], C4.5 [5], SPRINT [6] etc. In general terms, most of the algorithms have two distinct phases, a *building phase* and a *pruning phase* [3]. In the building phase, the training data set is recursively partitioned until all the instances in a partition have the same class. The result of this procedure is a tree that classifies every data item from the training set. However, the tree constructed may be sensitive to statistical irregularities of the training set. Thus, most of the algorithms perform a pruning phase after a building phase, in which nodes are pruned to prevent overfitting and to obtain a tree with higher accuracy.

The algorithms ID3 [3] and C4.5 [5] are based on a statistical property, called *information gain,* in order to select the attribute to be tested at each node in the tree. The measure definition is based on entropy used in information theory, “which characterizes the (im)purity of an arbitrary collection of examples” [3].

### Support Vector Machines (SMV)

Support Vector Machines provide a powerful methodology for solving problems in nonlinear classification.The theory of Support Vector Machines (SVMs) was first introduced by Vapnik and was developed from the theory of Structural Risk Minimization [7]. “SVMs learn the boundary regions between samples belonging to two classes by mapping the input samples into a high dimensional space, and seeking a separating hyperplane in this space. The separating hyperplane is chosen in such a way as to maximize its distance from the closest training samples” [8]. SVMs have been proved very effective for text categorization because of their property of learning independently of the dimensionality of the feature space [9]. Furthermore, Kivinen et al. [10] have proved both in theory and in practice that SVMs can handle problems with dense concepts and sparse instances, which are often seen in documents vectors. In addition, SVMs do not need parameter tuning, as they have an automatic way to find good parameters.

### k-Nearest Neighbor Learning (kNN)

All the above learning methods are called *eager methods* because they give an explicit description of the target function on the whole training set. On the other hand there are the instance-based methods which are called lazy methods, because the classification model is not built a priori. In their learning procedure they just store all the training instances and in the classification procedure they assign the target function to a new instance. The k-Nearest Neighbor learning (kNN) is the most basic of the instance-based methods. The kNN algorithm stores all available examples and classifies new instances of the example language based on similarity measure. The classification of an instance is most similar to the classification of other instances that are nearby in the vector space, usually in terms of the Euclidean distance [3].

### AIRS – A Resource Limited Immune Classifier

### Neural Networks

In machine learning and cognitive science, an **artificial neural network** (**ANN**) is a network inspired by biological neural networks (the [central nervous systems](https://en.wikipedia.org/wiki/Central_nervous_system) of animals, in particular the [brain](https://en.wikipedia.org/wiki/Brain)) which are used to estimate or [approximate](https://en.wikipedia.org/wiki/Universal_approximation_theorem) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) that can depend on a large number of [inputs](https://en.wikipedia.org/wiki/Argument_of_a_function) that are generally unknown. Artificial neural networks are typically specified using three things: [[MacKay, David, J.C.](https://en.wikipedia.org/wiki/David_J.C._MacKay) (2003). [*Information Theory, Inference, and Learning Algorithms*](http://www.inference.phy.cam.ac.uk/itprnn/book.pdf) (PDF). [Cambridge University Press](https://en.wikipedia.org/wiki/Cambridge_University_Press). [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [9780521642989](https://en.wikipedia.org/wiki/Special:BookSources/9780521642989)]

* **Architecture** specifies what variables are involved in the network and their [topological](https://en.wikipedia.org/wiki/Topology) relationships—for example the variables involved in a neural network might be the [*weights*](https://en.wikipedia.org/wiki/Weighting) of the connections between the[neurons](https://en.wikipedia.org/wiki/Artificial_neuron), along with *activities* of the neurons
* **Activity Rule** Most neural network models have short time-scale dynamics: local rules define how the *activities* of the neurons change in response to each other. Typically the activity rule depends on the *weights* (the parameters) in the network.
* **Learning Rule** The learning rule specifies the way in which the neural network's *weights* change with time. This learning is usually viewed as taking place on a longer time scale than the time scale of the dynamics under the activity rule. Usually the learning rule will depend on the *activities* of the neurons. It may also depend on the values of the *target values* supplied by a teacher and on the current value of the weights.

For example, a neural network for [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition) is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a [function](https://en.wikipedia.org/wiki/Function_(mathematics)) (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, the output neuron that determines which character was read is activated.

Like other machine learning methods – systems that learn from data – neural networks have been used to solve a wide variety of tasks, like [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), that are hard to solve using ordinary[rule-based programming](https://en.wikipedia.org/wiki/Rule-based_programming).

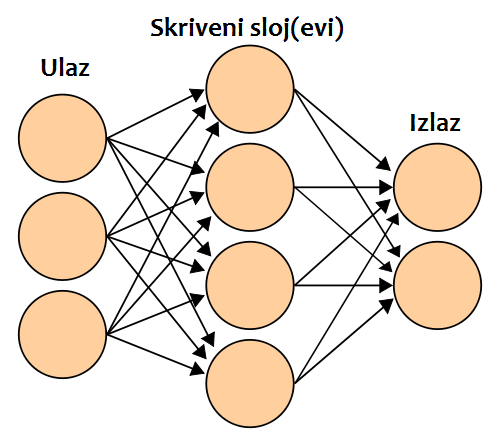
Deep learning models have achieved remarkable results in computer vision (Krizhevsky et al., 2012) and speech recognition (Graves et al., 2013) in recent years. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models (Bengio et al., 2003; Yih et al., 2011; Mikolov et al., 2013) and performing composition over the learned word vectors for classification (Collobert et al., 2011). Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close—in euclidean or cosine distance—in the lower dimensional vector space. Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features (LeCun et al., 1998). Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing (Yih et al., 2014), search query retrieval (Shen et al., 2014), sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011).

## Вештачке неуронске мреже

Вештачке неуронске мреже (ВНМ, енг*. Artificial Neural Netwoks – ANN*) опонашају биолошке механизме процесиранја информација. Типично су дизајниране да врше нелинеарно мапирање неког скупа улаза на скуп излаза. Са ВНМ покушаавју се постоћи перфомансе биолошких система користећи међусобне везе између једноставних процесних елемената аналогно биолошким неуронима. ВНМ уче на основу искуства и имају способност генерализације на основу претходно научених примера. Мењају своје понашање као одговор на околину и идеалне су у случајевима где потребни алгоритам мапирања није познат и где се захтева толерантност према неисправним/непотпуним улазним подацима [18 Jocic]. ВНМ нуде одређене предности у односу на неке конвенционалне приступе и алгоритме, а оне подразумевају способност генерализације, паралелизам, дистрибуирану меморију, редудантност и учење.

ВНМ се састоје од једноставних процесних елемената (неурона) који се повезују на одређени начин. Ова повезаност се може представити у форми усмереног графа где је сваки процесни елемент предствљен као чвор, а свака усмерена линија као веза (синапса) између два чвора. Неопходно је нагласити да је смер протока сигнала, односно информација означен смером стрелице на овој линији [19 Jocic]. Понашање обучене ВНМ зависи од тежина, које представљају јачину везе између два процесна елемента. Типична структура ВНМ са простирањем сигнала унапред (енг. feed forward мрежа). Је приказана на слици БРОЈ СЛИКЕ!. Први слој представња улазне неуроне, који преко синапси (веза између неурона) шаљу податке другом, односно скривеном слоју неурона, који их затим опет преко својих синапси шаљу излазном слоју. Сложенији системи имају више скривених слојева, као и повећан број неурона у улазном и излазном слоју, а постоје и архитектуре са повратном спрегом. Свака синапса има своју тежину, која представља параметар у манипулисању подацима током израчунавања.

Вредности тежина у везама између неурона су повезане са концептом обучавања неуронске мреже. Ово обучавање је заправо учење по примеру и представља вероватно најбитнији процес како би мрежа функционисала на жељени начин. Огромна предност ВНМ је да за решавање проблема није потребан добро дефинисан алгоритамски процес пресликавања улаза на излаз. Све што је потребно за већину ВНМ је скуп репрезентативних примера жељеног пресликавања, односно мапирања, на основу којих се мрежа прилагођава мењајући вредности тежина синапси. Додатно, ВНМ су робусне у смислу да ће вероватно дати задовољавајуће излазне вредности иако на улазу добије вредности које никад пре није „видела“ (нпр. улаз са шумом). Инхерентна способност да се носи са зашумљеним и нејасним шаблонима представља веома важну предност у односу на традиционална алгоритамска решења. Моћ приступа проблемима помоћу ВНМ не лежи обавезно у елеганцији самог решења већ у способности мреже да пронађе своје решење за одређене проблеме уз дате само прмере жељеног понашања. [Jocic 19]



Слика : Типична структура вештачке неуронске мреже

Вештачке неуронске мреже су нашле разноврсну примену у великом броју проблема, укључујући адаптивно управљање, оптимизацију, медицинску дијагностику, доношење одлука, као и у обради сигнала и информација, укључујући препознавање и обраду говора. ВНМ су се показале као веома погодне за процесирање сензорских података, посебно за издвајање обележја и аутоматско препознавање сигнала и вишедимензионалних објеката. Препознавање шаблона се, ипак, издвојило као највећа примена ВНМ јер структура мреже одговара задацима које биолошка неуронска мрежа обавља успешно, а препознавање шаблона је добар пример где биолошки системи надмашују традиционалне приступе рачунарске интелигенције [18]. Као специфичан облик примене препознавања шаблона издваја се и сематничка класификација текста која је и предмет овог рада.

### Основни елементи и појмови ВНМ

Као што је већ речено, вештачка неуронска мрежа се састоји од скупа једноставних процесних јединица које међусобно комуницирају слањем сигнала преко тежинских веза. Могу се издвојити одређени аспекти и елементи овог паралелног дистрибуираног модела [ Jocic 20]:

• скуп процесних јединица, односно неурона

• стање активације *yk* за сваку процесну једицину, које одговара вредности излаза те јединице

• везе између процесних јединица. Генерално свака веза је дефинисана тежином *wjk* која одређује ефекат сигнала јединице ј на јединицу *k*

• правило пропагације, које одређује ефективну вредност улаза ск јединице на основу њених спољашних улаза

• активациона функција *Fk*, која одређује нови ниво активације на основу ефективног улаза *sk(t)* и тренутног нивоа активације *yk(t)*

• праг активације *ʘk* за сваку процесну јединицу

• метод скупљања информација, односно правило обучавања

• окружење унутар кога систем ВНМ функционише и које генерише улазне сигнале и уколико је потребно и сигнале грешке

### Обучавање ВНМ

Како би неуронска мрежа за одређене скупове улазних података генерисала жељене излазне вредности, непходно ју је претходно конфигурисати, односно подесити везе између неурона. Постоји различити поступци за подешавање јачина веза, односно тежина између неурона. Један начин је подесити тежине експлицитно, користећи *a priori* знање. Други начин подразумева “обучавање” неуронске мреже тако што јој се саопшти обучавајући скуп података и изврше промене у тежини према неком правилу обучавања.

Постоје два главна начина обучавања неуронских мрежа:

• *Надгледано или асоцијативно учење* – ВНМ се обучава улазним вредностима и њима одговарајућим, односно жељеним излазним вредностима. Ове парове улазно-излазних вредности може обезбедити неки екстерни учитељ, или систем који садржи саму мрежу (самонадгледано учење).

• *Ненадгледано учење или самоорганизација* – излазне процесне јединице се обучавају да одговарају на кластере шаблона у улазним подацима. Систем би сам требао да открије статистички истакнуте особине у улазним подацима. За разлику од надгледаног учења, овде не постоји *a priori* скуп катеогрија у који шаблони треба да буду класификовани, већ систем сам мора развити своју репрезентацију улазних стимулуса.

### Архитектуре, структуре и топологије ВНМ

На основу начина на који су процесни елементи повезани и на који начин информације пропагирају кроз мрежу, разликују се две основне топологије ВНМ:

• ВНМ са *простирањем сигнала унапред* (енг. *feed-forward*) – где је ток података од улаза ка излазу стриктно једносмеран. Процесирање података може бити обављано преко више слојева процесних јединица, али не постоје повратне везе, односно не постоје везе које повезују излазе процесних јединица на улазе других процесних јединица у истом или претходном слоју. Примери ове мреже су једнослојни перцептрон, вишеслојни перцептрон и кохоненова мапа. У овај тип мрежа спадају и конволутивне мреже .

• *Рекурентне* ВНМ – за разлику од *feed-forward* мрежа, имају повратне везе и динамичке особине ових мрежа су веома важне. Примери ове мреже су класична рекурентна мрежа, Хопфилдова мрежа и Кохоненова мапа.

#### Једнослојни перцептрон

#### Једнослојни перцептрон је једнослојна феед-форwард мрежа, односно има само улазни и излазни слој, без скривених слојева и без повратних веза. Први га је увео Ф. Росенблатт 1958 године [21]. Име “једнослојни” се односи заправо на излазни слој, односно на само један слој веза од улаза до излаза, иако не постоје скривени међуслојеви. Дакле, ово је једноставна неуронска мрежа која често има бинарне вредности улаза и излаза (0 или 1). Обучава се надгледаним процесом обучавања и ова мрежа може да решава основне логичке проблеме, као што су И (АНД) или ИЛИ (ОР). Такође се користи и у сврхе класификације шаблона. Компликованије логичке операције, као што је ексклузивно ИЛИ (XОР) не могу бити решене помоћу једнослојног перцептрона.

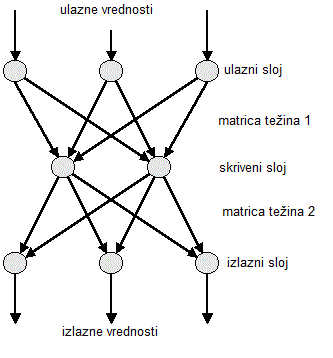
#### На слици 2.6 је дат пример једнослојног перцептрона са 3 улазна неурона и 2 излазна неурона и њиховим међусобним везама.

#### 

Slika 2.6: Primer jednoslojnog perceptrona

#### Вишеслојни перцептрон

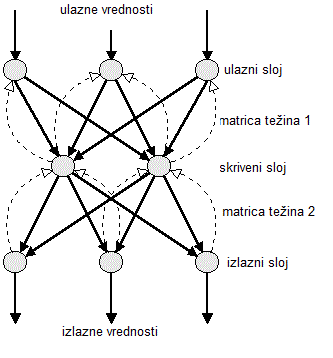
#### Вишеслојни перцептрон представља вишеслојну *feed-forward* мрежу, односно то је проширен једнослојни перцептрон који има један или више слојева неурона између улазног и излазног слоја. Увели су га M. Minsku i P. Pappert 1969. године [22]. Због своје проширене структуре, вишеслојни перцептрон може да решава сложеније проблеме, као и сложеније логичке операције, укључујући и ексклузивно ИЛИ (XОР). Оваква ВНМ се обучава надгледаним процесом обучавања.



Слика ??: Пример вишеслојног перцептрона

#### Рекурентна мрежа

#### Рекурентна ВНМ има сличну структуру као вишеслојни перцептрон уз додатак повратних веза, и користи надгледани backpropagtion алгоритам за обучавање и једна је од најмоћнијих врста неуронских мрежа. Увели су је G.E. Hinton, E. Rumelhart i P.J. Williams 1986. године [23] и највише се користи за решавање сложених логичких операција, класификацију шаблона и анализу говора. Пример рекурентне мреже је приказан на слици 2.8.



Слика ?? Пример рекурентне неуронске мреже

#### Kohonenova mapa ?

#### Hopfildova mreza ?

### Конволуционе вештачке неуронске мреже

Конволуциона неуронска мрежа (КНМ, енг. Convolutional Neural Network-CNN) је тип *feed-forward* вештачке неуронске мреже у којој је шаблон повезаности међу неуронима инспирисан организацијом животињског визелног кортекса, чији су индивидуални неурони уређени на такав начин да одговарају преклапању региона који поплочавају поље вида [ *["Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation"](http://deeplearning.net/tutorial/lenet.html)*.DeepLearning 0.1. LISA Lab*. Retrieved31 August 2013].* Конволуционе мреже су иснпирисане биолошким процесима и варијацијама вишеслојних перцептрона дизајнираних тако да користе минимално препроцесинга. Ове мреже се примењују у различитим областима као што су препознавање слика и видеа, системи препоруке, обрада природних језика и семантичка класификација. -

Када се користе за препознавање слика, КНМ се састоје од више слојева мањих колекција неурона који обрађују делове улазне слике, те колекције се зову рецептивна поља. Излази ових поља се онда склапају тако да се њихови улазни региони преклапају како би се добила боља репрезентација оригиналне слике. Овај поступак се понавља за сваки слој. Склапање омогућава КНМ да толерише транслације улазне слике. [isto referenca kao iznad]

Конволуционе мреже могу укључивати локалне или глобалне pooling слојеве, који комбинују излазе неуронских група (кластера). Такође, могу се састојати од разноликих комбинација конволуционих и потпуно повезаних слојева. Како би се смањио број слободних параметара и повећала генерализација, уводи се конволуциона операција на малих улазним регионима. Велика предност конволуционих мрежа је употерба дељене тежине у конволуционим слојевима, што значи да се исти филтер (тежинска банка ? ) користи за сваки пиксел у слоју смањујући значајно меморијску захтевност и побољшавајући перформансе. [*"Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation"*](http://deeplearning.net/tutorial/lenet.html).DeepLearning 0.1. LISA Lab*. Retrieved31 August 2013]*

#### Историјат

Дизајн конволуционих неуронских мрежа прати откриће визуелних механизама код живих организама. Још 1968. год. показано је да животињски визални кортекс задржи комплексно уређење ћелија одговорнних за детектовање светла у малим, преклапајућим подрегионима поља вида, званих рецептивна поља. Тада су дифренциране две врсте основних ћелија: једноставне и сложене. Једноставне реагују максимално на специфичне граничне шаблоне у свој рецептивном пољу, а сложене које поседују веће рецептивно поље и делују као локални филтери.

ВНМ неуронска мрежа, неокогнитрон представља предак КНМ. Настала је 1980. год. и разликује се од КНМ по томе што не приморава јединице лоциране на различитим позицијама да имају исте тренирајуће тежине. Ова идеја се појављује у 1986. у раду о backpropagation алгоритму. Њихов дизајн је унапређен 1998. , а генерализован 2003. Једна од најпознатијих имплементација ових мрежа је LaNet-5 мрежа која може да класификује цифре и која се користи за препознавање ручно написаних чекова. Међутим, када се ове мреже покушају применити на комплекснијим проблемима, димензије мреже се повећавају и саме перфомансе се драстично погоршавају због лимитираних рачунарских ресурса.[ LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning".]

Почевши од 2005. год. многобројни радови предлажу и приказују имплементације КНМ мрежа тренираних на GPU уређајима. Овакве имплементације показују импресивне резултате.

#### Специфичности КНМ

Иако су традиционални вишеслојни перцептрони успешно коришћени за препознавање слика, због потпуне повезаности чворова нису се добро скалирале на слике високе резолуције. Овакве архитектуре мрежа у обзир нису узимале просторну структуру података, третирајући везе између удаљених и блиских пиксела на потпуно исти начин. Јасно је да је потпуна повезаност неурона беспотребна и да велики број параметара брзо доводи до претренирања (енг. overfitting).

Конволуционе мреже су инспирисане варијацијама вишеслојних перцептрона дизајнираних да опонашају животињски визуални кортекс. Ови модели ублажавају проблеме са којима се суочавају вишеслојни перцептрони користећи јаку просторну повезаност присутну у сликама. За разлику од вишеслојних перцептрона КНМ поседују следеће специфичности:

* **3D уређени неурони**. Слојеви КНМ имају неуроне уређене у три димензије. Неурони у слоју су повезани само са малим регионим претходног слоја, тај регион се зове рецептивно поље.
* **Локална повезаност**. У складу са концептом рецептивних поља, КНМ користе локалну просторну повезаност тако што намећу шаблон повезаности између неурона суседних слојева. На тај начин архитектура осигурава да научени „филтери“ најаче реагују на локалне (у простору) улазне шаблоне.
* **Дељене тежине**. У КНМ, сваки филтер се понавља дуж читавог видног поља. Те поновљене јединице деле исте параметре и формирају мапу неке особине (енг. feature map). У пракси то значи да сви неурони у неком слоју детектују исту особину, тј. мрежа је способна да препозна исту особине без обзира где се она налази на улазној слици, просторној репрезентацији улаза.

#### Градивни елементи

КНМ се формирају кроз низ различитих слојева који траснформишу улазне величине у излазне. Постоје више различитих слојева који се могу користити и они су набројани испод:

* **Конволуциони слој**. Основни градивни елемент КНМ. Параметри слоја се састоје од сет филтера (кернела) који имају способност учења. Филтери поседују мало рецептивно поље. Мрежа учи „филтере“ да реагију на специфичне типове особина на неком делу улазног простора. Конволуциони слој користи локалну повезаност неурона, насупорт тоталној повезаности. Три хеперпараметра одређују величину излаза конволуционог слоја, и то су:
  + **Дубина (енг. depth)** – Одређује број неурона који су повезани на исти регион улаза.
  + **Корак (енг. stride) –** Одређује степен преклапања при повезивању неурона на улаз.
  + **Нула-одсецање (енг. zero-padding) –** Даје могућност задржавања једнакости величина улаза и излаза, која се могла изгубити због преклапања.

Величина излазног слоја може бити израчуната као функција величине улаза ***W***, величине кернела (филтера) конволуционог слоја ***К***, корака ***S*** i zero-padding-a ***P***. Формула гласи: ***(W-K+2P)/S + 1.*** Улазне вредности морају бити тако подешене да резултат ове формуле буде цео број. Такође, битна карактеристика Конволуционог слоја је и *дељење параметара (енг. parameter sharing) .* Шема дељења параметара је уведена како би се контролисао број слободних параметара.

* ***Pooling* слој.** *Pooling*је још један битан концепт у КНМ и представља вид нелинеарног поједностављења. Постоји више функција које омогућавају *pooling,* али најпознатија и највиеше у употребије *max-pooling.* Она дели улаз на непреклапајуће правоугаонике и као излаз из сваког узима његову максималну вреност. Пример *max-pooling* функције дат је на слици *??.* Образложење за кориршћење *Pooling-а* лежи у томе да није потребно знати егзагтну локацију пронађене особине, већ је довољна и информација са мањом прецизношћу. На овај начин смањује се број параметара и ресурсна захтевност мреже, а токође се контролише и проблем *overfitting-a.*



**Слика ?? max-pooling функција**

* ***ReLU* слој.** *ReLU (*скраћеница од енг. *Rectified Linear Units)* је слој неурона који примењује незасићујућу активациону функцију ***f(x) = max(0,x)****.* Овај слој повећава нелинеарност функција одлуке целокупне мреже.
* ***Потпуно повезани слој.*** Овај слој следи после више конволуционих и *max-pooling* слојева. Он обавља високо нивовско одлучивање.
* ***Слој губитака (енг. loss layer).*** Овај слој специфицира како се тренирање мреже односи са разликама између предвиђених и правих излаза и обично је последњи слој у мрежи.Постоји више функција које се могу користити у овом слоју. Неке од њих су *softmax, sigmoid cross-entropy* и *Euclidean loss.*

#### Избор хиперпараметара

КНМ користе више хиперпараметара него стандардне ВНМ. Уобичајена правила за брзине учења и регуларизацију се могу применити и на КНМ, али треба имати у виду следеће смернице приликом оптимизације КНМ.

* **Број филтера** – Са обзиром да се мапа особина смањује са дубином мреже, слојеви ближи улазу често имају мање филтера од оних који су даље. Како би се очувале информације пожељно је одржавати број активација (производ мапа особина и броја пиксела, или улазних позиција) тако да се не смањује. Број мапа особина директно контролише капацитет и зависи од броја доступних примера и комплексности задатка.
* **Облик филтера –** Постоји доста варијација облика филтера и избор облика директно зависи од улазног скупа података тј. *dataset-*a.
* **Облик *max-polling*-a –** Обично 2x2. Већи *max-polling* облици обично са собом носе и већи губитак података.

#### Методе регуларизације

Методе регуларизације се углавном баве решавањем проблема претренирања (енг. *overfitting).* Постоје две основне групе метода, а то су емпиријске и експлицитне. У емпиријске спадају *dropuout, dropoutConnect, stochastic pooling, artificial data*, а у ескплицитне рано заустављање (енг. *early stopping),* смањење броја параметара, опадање тежина (енг. *weight decay)*, L2 регуларизација, L1 регуларизација и ограничање максималне норме (енг. *max norm constraints).[* Srivastava, Nitish; C. Geoffrey Hinton; Alex Krizhevsky; Ilya Sutskever; Ruslan Salakhutdinov (2014). ["Dropout: A Simple Way to Prevent Neural Networks from overfitting"](http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf)]

#### Примене

КНМ налазе примену у много различитих сфера, а због својих специфичности нове примене се свакодневно проналазе.

* **Препознвање слика** ­– КНМ се често користе у препознавању слика, оне имају остварену грешку од 0,23% на MINST бази података, што је у фебруару 2012. Године била најнижа грешка на тој бази података. Много папири говоре у прилог томе да су КНМ изузетно погодне за класификацију слика и да се „изненађујуће“ брзо тренирају.[ Ciresan, Dan; Meier, Ueli; Schmidhuber, Jürgen (June 2012). ["Multi-column deep neural networks for image classification"](http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6248110)]
* **Видео анализа –** Видео анализа је доста комплексина од препознавања слика, због додатне димензије – временске. Ипак неке КНМ проналазе примену и у овој сфери, тако што врше конволуције и по времену и по простору.[ accouche, Moez; Mamalet, Franck; Wolf, Christian; Garcia, Christophe; Baskurt, Atilla (2011-11-16). "Sequential Deep Learning for Human Action Recognition"]
* **Обрада природних језика –** КНМ модели показују се као ефикасни за различите проблеме у сфери обраде природних језика и остварују одличне резултатеу задацима као што су семантичко парсирање, моделирање реченица, класификација, предикција.[ Grefenstette, Edward; Blunsom, Phil; de Freitas, Nando; Hermann, Karl Moritz (2014-04-29). "A Deep Architecture for Semantic Parsing"]

КНМ налазе примену и у областима као што су проналажење опојних средстава, симулирање игара (као нпр. игра GO).[Clark, Christopher; Storkey, Amos (2014). "Teaching Deep Convolutional Neural Networks to Play Go".] [Wallach, Izhar; Dzamba, Michael; Heifets, Abraham (2015-10-09). "AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery"]

## Обрада природних језика

Обрада природног језика (ОПЈ) је област вештачке интелигенције и лингвистике која се бави проучавањем проблема аутоматског произвођења и разумевања природних људских језика. Системи за произвођење природног језика претварају информације из рачунарске базе података у људски језик који природно звучи а системи за разумевање природног језика претварају примере људског језика у више формалне представе са којима рачунарски програми лакше манипулишу.

### Историјат

Историјат ОПЈ генерално почиње у педесетим годинама прошлог века, иако се неки радови јављају и раније. Алан Тјуринг 1950. год. издаје чланак под називом "Computing Machinery and Intelligence" који предлаже, сада већ чувени „Тјурингов тест“ као критеријум интелигенције. Експеримент познат као „Georgetown experiment“ 1954. год. укључује аутоматски превод више од 60 реченица на руском на енглески језик. Аутори овог ескперимента су тврдили да ће се проблем аутоматском превођења решити кроз пар година, ипак прогрес је текао много спорије и застој у финансирању пројеката који се баве аутоматским преводом присутан је све до осамдесетих година прошлог века.[Hutchins, J. (2005). ["The history of machine translation in a nutshell"](http://www.hutchinsweb.me.uk/Nutshell-2005.pdf) ]

Током шездесетих година јавља се више ОПЈ система вредних помена. SHRDLU представља систем који ради са ограниченим блоковима речи у ограниченим речницима, а ELIZA представња симулацију психотерапеута који је имао назнаке „скоро“ људске интеракције иако није имао уграђено знање о људској мисли или емоцијама. Тај систем је био написан од стране Joseph Weizenbaum –а.

Седамдесетих година прошлог века многи програмери су писали концепуталне онтологије које су се бавиле структуирањем информација из стварног света у податке разумљиве рачунарима. Примери ових система су *MARGIE (Schank, 1975), SAM (Cullingford, 1978), PAM (Wilensky, 1978), TaleSpin (Meehan, 1976), QUALM (Lehnert, 1977), Politics (Carbonell, 1979),* и *Plot Units (Lehnert 1981).* Током овог времена настали су и „роботи за дописивање“ као што су *PARRY*, *Racter*, и *Jabberwacky*.

До 1980-их, већина ОПЈ система су била базирана на комплексном скупу ручно писаних правила. Крајем те деценије догађа се револуција у ОПЈ увођењем алгоритама машинског учења у сврху обраде језика. Ова револуција настаје као последица повећавања рачунарске моћи и слабљења доминације лингвистичке теорије Чомског која је обесхрабривала машинско учење као приступ обради језика. Рани алгоритми машинског учења, као стабла одлучивања, креирали су системе са „ако-онда“ правилима сличним пређашњим ручно писаним правилима. Међутим, употреба скривених Марковљевих модела и истраживања статистичких модела креирају системе са пробабилистичким одлукама базираним на придруженим процењеним вредностима додељеним неким особинама улаза. Примери таквих система су *cache* модели језика на којима почивају многи модерни системи за препознавање говора. Овакви системи приказују већу робустност на непознате улазе и улазе са грешкама.

Многобројни рани успеси везани су за поље машинског превода, захваљујући истраживању IBM-a, где су са успехом креиране све комплекснији статистички модели. Ови модели су могли да користе постојеће вишејезичке скупове текста који су настали као плод рада Парламента Канаде и Европске Уније. Ипак доста других покушаја имплементације машинског учења у сврху ОПЈ наилазило је на мањак доступних података за тренирање, те се доста времена посветило унапређењу метода учења са лимитираном количином података.

Скорашња истраживања се највише фокусирају на ненадгледане и семи-надгледане алгоритме учења. Такви алгоритми су у стање да уче из података који нису ручни анотирани (нпр. сортирани по класама) или из података који садрже мешавину анотираних и неанотираних података. Овакве алгоритми дају генерално резултате слабије прецизности, но са друге стране постоји огромна количина неанотираних података (нпр. на интернету) која може надогнадити слабију прецизност. [ *["SEM1A5 - Part 1 - A brief history of NLP"](http://www.cs.bham.ac.uk/~pjh/sem1a5/pt1/pt1_history.html)*]

### Употреба машинског учења

Модерни ОПЈ алгоритми су базирани на машинском учењу, поготово на статистичком машинском учењу. Парадигма машинског учења се разликује од већине пређашњих покушаја обраде језика. Пређашње имплементације обраде језика су обично укључивале директно и ручно кодиране сетове правила. Парадигма машинског учења инсистира на коришћењу уопштених алгоритама учења, често, али не и увек, базираних на статистичком закључивању(енг. *statistical inference*). Такво закључивање се базира на учењу правила анализом великог корпуса примера из стварног света. Корпус представља сет докумената са анотираним вредностима из који алгоритам „учи“.

Системи базирани на алгоритмима машинског учења поседују многе предносту у односу на системе са ручно написаним правилима:

* Процедуре учења током машинског учења се аутоматски фокусирају на најчешће случајеве, док са ручно написаним правилима често није очигледно на које случајеве треба обратити посебну пажњу.
* Аутоматске процедуре учења могу искористити статистичко закључивање како би креирали моделе који су робустни према непознатим и улазима са грешкама. Креирање такве робусности помоћу ручно писаних правила је изузетно тешко.
* Системи базирани на аутоматском учењу се могу направити прецизнијим са једноставним повећавањен броја улазних података, за разлику од система са ручним правилима чији се резултати могу поправити само унапређењем тих правила.

Област ОПЈ која се бави приступима учењу назива се Учење Природних Језика (УПЈ, енг. *natural language learning* *NLL*). УПЈ се преклапа са облашћу рачунарске психолингвистике (енг. *computational psycholinguistics*). [http://www.signll.org/conll/]

### Примене

У наставку следи листа најчешћих истраживаних задатака којима се ОПЈ може бавити. Неки од наведених задатака имају примену у реалним ситуацијама, док други се користе само као помоћ при решавању комплекснијих проблема. Наведени не само што иза себе имају доста истраживачког рада, већ је њихова проблематика добро дефинисана, дефинисани су критеријуми евалуације, постоји корпус над којим се може тестирати специфична имплементација и такмичења посвећена специфичном задатку.

* **Аутоматска сумаризација** – Циљ је аутоматски генерисати сажетак неког дела текста, тако да он буде читљив и разумљив.
* **Решавање кореференци** – Циљ овог проблема је одредити где и да ли се нека реч помиње у неком другом делу датог текста. Пример овог проблема је повезивање заменица са именицама на које се оне односе у неком тексту.
* **Анализа дискурса** – Овај проблем садржи доста повезаних задатака. Један од њих је и идентификација структуре дискурса или односа међу реченицама (елаборација, објашњавање или контаст).
* **Машинско превођење** – Аутоматски превод са једног људског језика на други. Ово је један од најтежих ОПЈ проблема коме је посвећено доста истраживачких ресурса.
* **Морфолошка сегментација** – Подела речи у засебне морфеме и њихова класификација. Комплексност овог задатка директно зависи од комплексности морфологија језика који је предмет истраживања.
* **Препознавање именованих ентитета** – Циљ овог задатке је одредити који ентитети из датог текста представљају властита имена, људи или места нпр., а одредити тип датог имена( особа, локација, организација нпр.).
* **Генерисање природног језика** – Претварање информација из рачунарских база података у језик читљив човеку.
* **Размевање природног језика** – Претварање делова текста у више формалне репрезентације, као што је логика првог реда, како би се њима лакше рачунарски манипулисало. [Yucong Duan, Christophe Cruz (2011), [*Formalizing Semantic of Natural Language through Conceptualization from Existence*](http://www.ijimt.org/abstract/100-E00187.htm) *]*
* **Оптичко препознавање карактера (енг. *Optical character recognition -OCR*)** – Одређивање текста преко слике одштаманог текста.
* **Парсирање** – Одређивање стабла парсирања за дату реченицу. Граматика природних језика је често двосмислена и обичне реченице често имају више могућих анализа.
* **Одговарање на питања** – Одредити одговор на питање написано на обичном-људском језику. Питања наравно могу варирати у комплексности и новоу апстракције. ["[Versatile question answering systems: seeing in synthesis](https://www.academia.edu/2475776/Versatile_question_answering_systems_seeing_in_synthesis)", Mittal et al., IJIIDS, 5(2), ]
* **Одређивање односа (енг. *Relationship extraction*)** – На датом тексту је потребо одредити односе међу именованих ентитета (нпр. „ко је коме родитељ?“ ).
* **Анализа сентимента (енг. *Sentiment analysis*)** – Odreђивање поларитета неког дела текста, као што је филмска критика и слично, тј. да ли је текст позитиван , негативан или неутралан).
* **Препознавање говора** – Одређивање текстуалне репрезентације неког звучног записа. Предстаља изузетно тежак проблем за чије решење је потребно користити велики број других задатака којима се бави ОПЈ, као на пример сегментација говора.
* **Сегментација говора** – Звучни запис говора неког или више људи потребно је поделити на речи.
* **Сегментација и препознавање тема** – Дати текст је потребно поделити у делове на основу препознатих тема о којима ти делови говоре.
* **Решавање проблема двосмислености речи (енг**. ***Word sense disambiguation)***- Oд више могућих значења речи потребно је изабрати ону која највише одговара контексту.
* **Кореновање (енг**. ***Stemming)*** – Представља процес редукције речи на њену основу или корен. Та основа или корен не мора бити једнака морфолошком граматичком корену речи. [Lovins, Julie Beth (1968). "Development of a Stemming Algorithm". *Mechanical Translation and Computational Linguistics*.]

## Твитер

### Опис

Твитер је онлајн друштвена мрежа која омогућава корисницима да постављају и читају кратке, до 140 карактера дуге, поруке зване твитови. Регистровани корисници могу да читају и постављају твитове док они нерегистровани могу само да читају. Корисници приступају твитеру путем веб странице или мобилних апликација. Седиште компаније је у Сан Франциску и постоји преко 35 канцеларија широм света. Лого твитера је приказан на слици ??!. Твитер поседује:

* Преко 300 милиона активних корисника
* Преко милијарду јединствених посета месечно
* Око 4000 запослених
* Подржава преко 40 језика
* Око 80% корисника је лоциране ван граница САД-а
* Приход од 2.21 милијарде и нето добит од 521 милион (2015. год.)

Слика ??! Лого твитера.

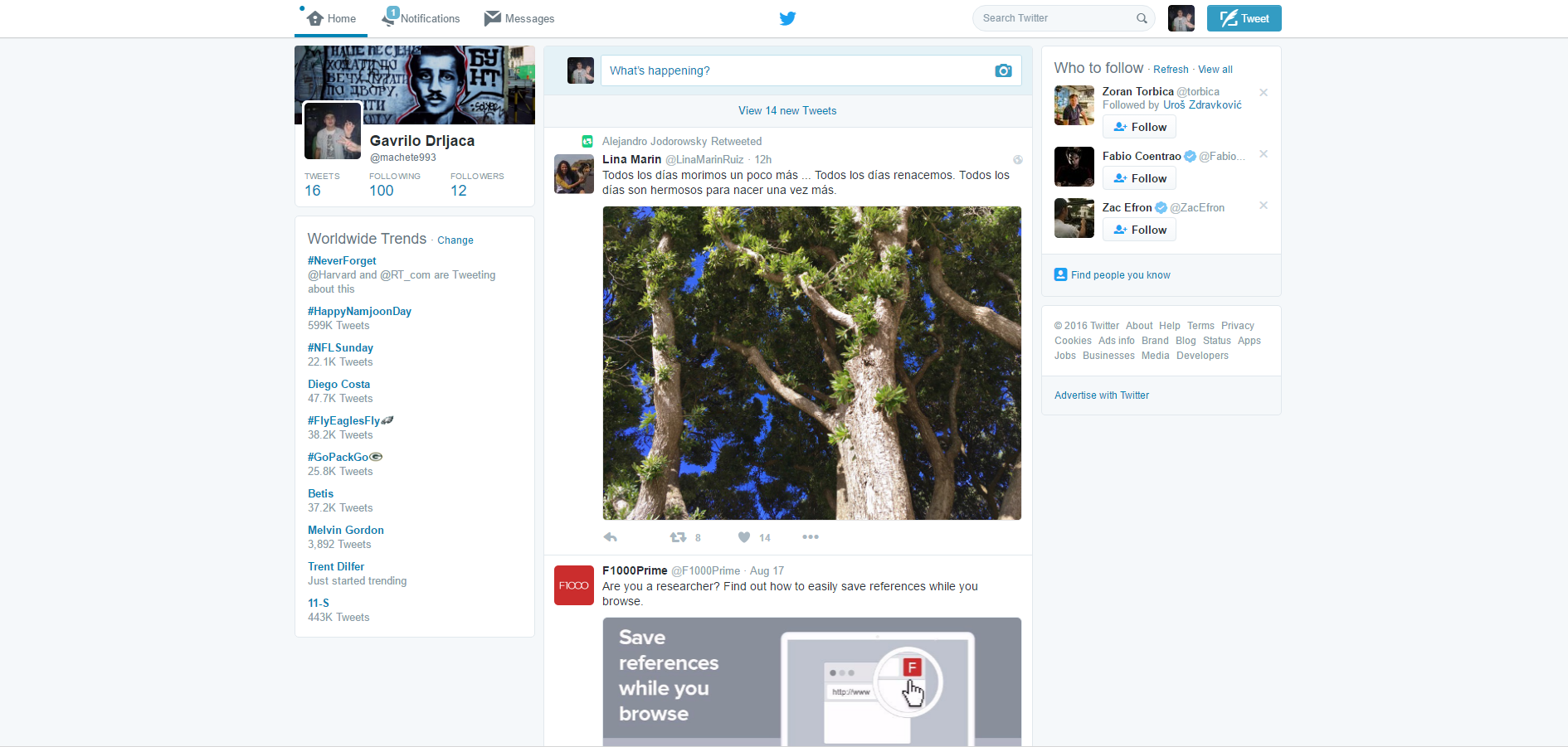
Твитер је креиран од стране  *Jack Dorsey, Evan Williams, Biz Stone* и *Noah Glass* 2006. год. и постао је доступан у јулу исте године. Сервис брзо добија глобалну популарност, са преко 100 милиона корисника и 340 милиона твитова по дану 2012. год. У 2013-ој години постаје једна од десет најпосећенијих сајтова на свету. [["Top Sites"](http://www.alexa.com/topsites). [Alexa Internet](https://en.wikipedia.org/wiki/Alexa_Internet). Retrieved May 13, 2013. ]

Твитови су видљиви од стране свих, али могуће је послати твитове који ће бити видљиви само од стране људи које „прате“ тај налог. Корисници се могу претплатити да твитове других људи, познатих као они које пратиш „*following”,* а они који су се претплатили на твитеве су познати као „*followers”.* Појединачни твитови се могу проследити другим корисницима „retweet” или „лајковати“ (од енг. like - свидети). Аутор Steven Johnson је 2009. год. објаснио основну механику Твитера као „изненађујуће једноставну“:

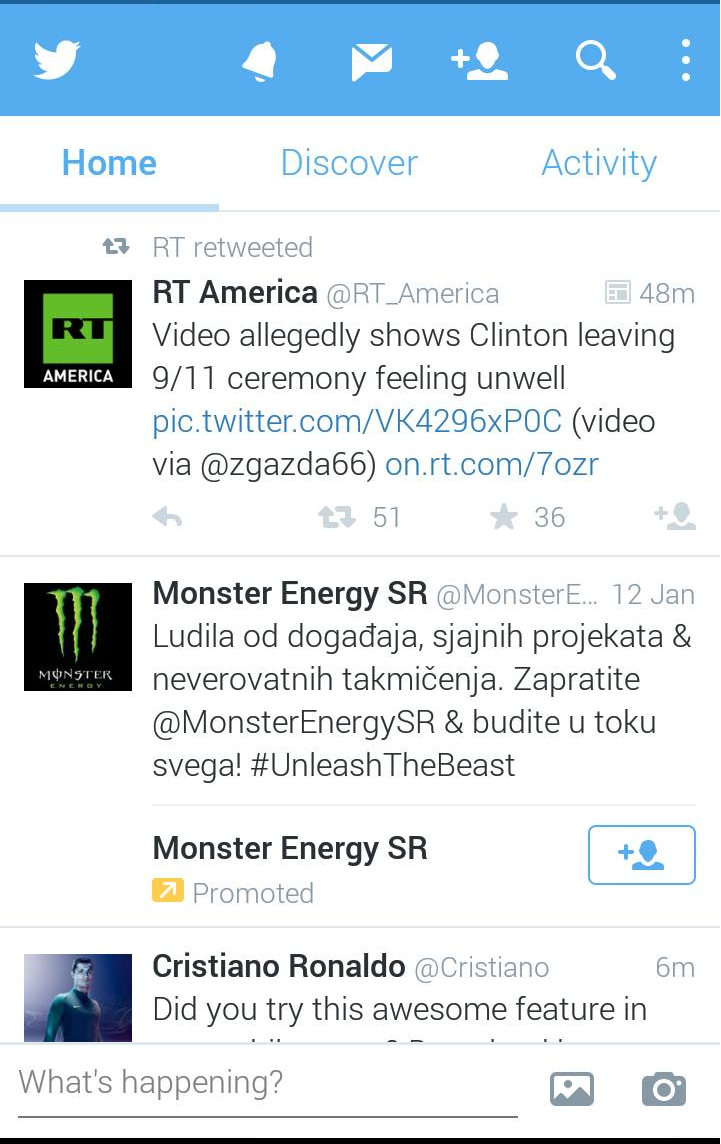
*Као друштвена мрежа, Твитер се базира на принципу пратилаца (followers). Када одаберете да пратите другог Твитер корисника, твитови тог корисника се појављују у обрнутом временском редоследу на вашој почетној страници. Акао пратите 20 људи, видећете микс твитова скролујући по почетној страници, занимљиве нове линкове, музичке препоруке, чак и размишљања о будућим едукацијама.* [[Johnson, Steven](https://en.wikipedia.org/wiki/Steven_Berlin_Johnson) (June 5, 2009).  ["How Twitter Will Change the Way We Live"](http://www.time.com/time/printout/0,8816,1902604,00.html). [*Time*](https://en.wikipedia.org/wiki/Time_(magazine)). RetrievedFebruary 13, 2011.]

Имплементација твитера у великој мери почива на *open-source* софтверу. Твитеров веб интерфејс користи *Ruby on Rails* фрејмворк, изграђен на побољшаној *Ruby Enterprise Edition* имплементацији *Ruby програмског* језика, који нуди боље перформансе. У почетку твитови су чувани у *MySQL* бази података, али због проблема са перформансама насталих због повећаног броја корисника, прелази се на *Java* сервер, под називом *Blender.* [["Twitter Search Is Now 3x Faster"](https://blog.twitter.com/2011/twitter-search-is-now-3x-faster). April 6, 2011 ]. У почетку за руковање порукама био је задужен *Ruby* сервер под називом *Starling,* али од 2009. год. постепено се врши миграција на софтвер написан на програмском језику *Scala.* Та миграција је донела значајно унапређење перформанси, те је било могуће подржати и између десет и двадесет хиљада захтева по хосту, за разлику од пређашњих двеста до триста. Развој Твитера иде од монолитне апликације ка архитектури повезаних сервиса који обављају засебне задатке. Сервис такође и пружа свој интерфејс путем кога је могућа комуникација са другим сервисима и апликацијама.

Интерфејс твитера је претрпео многобројне промене током свог постојања. Априла 2009. год. додато је поље за претраге и додатно поље са „*trending”* темама', тј. фразама које се најчешће јављају у порукама. Све поруке су тренутно индексиране те се може рећи да је Твитер добио поприлично важну улога, сервиса који омогућује тренутно информацију о најважнијим тренутним дешавањима. Изглед тренутног веб интерфејса Твитера је на слици ??!, а интерфејс Твитер мобилне, у овом случају андроид, апликације је на слици ??. [[Stone, Biz](https://en.wikipedia.org/wiki/Biz_Stone) (April 30, 2009). ["Twitter Search for Everyone!"](http://blog.twitter.com/2009/04/twitter-search-for-everyone.html). Twitter. Retrieved May 7, 2008.]



Слика ??!. Изглед веб интерфејса Твитера.



Слика ??!: Изглед интерфејса Твитерове мобилне апликације.

### Структура и особености твитова

Твитови представљају кратке текстуалне поруке дужине до 140 карактера. Због своје ограничене дужине честе у себи садрже емотиконе, скраћенице или сленг. Таква структура твитова представља додатни изазов њиховој анализи методама машинској учења.

Твитови могу да садрже ознаке ([енг.](https://sr.wikipedia.org/wiki/%D0%95%D0%BD%D0%B3%D0%BB%D0%B5%D1%81%D0%BA%D0%B8_%D1%98%D0%B5%D0%B7%D0%B8%D0%BA) *tags*) састављене од појма са префиксом # ([енгл.](https://sr.wikipedia.org/wiki/%D0%95%D0%BD%D0%B3%D0%BB%D0%B5%D1%81%D0%BA%D0%B8_%D1%98%D0%B5%D0%B7%D0%B8%D0%BA) *hashtag*), попут #serbia. На овај начин се омогућава проналажење твитова на одређену тему једноставним тражењем употребљене ознаке. Знак @ испред корисничког имена, попут @корисничко\_име, користи се за упућивање реплике одређеном кориснику, или више њих. Твитове који почињу са @корисничко\_име могу видети сви остали корисници, али се такве поруке сматрају директном репликом кориснику чији је надимак први у низу. За упућивање директне (приватне) поруке неком кориснику користи се мало латинично слово *d* испред корисничког имена особе којој се порука шаље (нпр: *d корисничко\_име порука*). Овако послате твитове може видети само особа којој је порука послата.

## Програмски језик Phyton

### Историјат

### Карактеристика и филозофија

### Синтакса и семантика

### Библиотеке (Керас итд.)

# Литература

1. http://www.canoo.net/services/OnlineGrammar/Wort/Nomen/Bedeutung/
2. Usama M. Fayyad, Gregory Piatesky-Shapiro, Padhraic Smuth and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press 1996
3. T. Mitchell. *Machine Learning.* McGraw-Hill, 1997
4. D. Michie, D. J. Spiegelhalter & C. C. Taylor. *Machine Learning, neural and statistical classification,* (edited collection). New York: Ellis Horwood, 1994
5. J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993
6. J. Shafer, R. Agrawal, M. Mehta. “*SPRINT: A scalable parallel classifier for data mining“.* In Proc. Of the VLDB Conference, Bombay, India, September 1996