Алгоритъма на Мислещата машина

Abstract

В тази статия разглеждаме въпросите „Какво е ИИ?“ и „Как да напишем програмата, която удовлетворява дефиницията за ИИ?“ Разглеждат се основните концепции и модули, които трябва да са в основата на тази програма. Най-интересната концепция, която се разглежда тук, е концепцията на абстрактните сигнали. Всеки от тези сигнали е свързан с резултата от провеждането на определен експеримент. Абстрактният сигнал е функция, които за всеки момент от време връща вероятността този експеримент да върне истина. Въвеждаме и понятието отворен експеримент, което е обобщение и отговаря на множество от затворени експерименти.

Въведение

През целия си живот съм се опитвал да си отговоря на въпроса „Какво е ИИ?“ и да напиша програмата, която е ИИ. От 16 години вече знам отговора на въпроса „Какво е ИИ?“, но алгоритъма на ИИ все ми се изплъзваше. Появяваха се отделни фрагменти, но все нещо липсваше, за да бъде сглобен целия пъзъл. Най-накрая събрах всички липсващи парчета и вече мога да ви представя този алгоритъм. Тоест, в тази статия ще намерите достатъчно подробно описание на алгоритъма на ИИ. Това звучи толкова гръмко, че вие едва ли ще ми повярвате. Честно казано дори и аз не си вярвам напълно. Ще повярвам чак когато някой напише програмата изпълняваща този алгоритъм и видя, че тази програма действително работи. Дори и да не повярвате в значимостта на тази статия, все пак се надявам тя да ви хареса.

Тук първо ще се занимаем с въпроса „Какво е ИИ?“ На този въпрос Алан Тюринг вече е отговорил още преди 66 години [5]. Към дефиницията на ИИ има и една малка добавка направена от мен през 2000 г. След това ще опишем задачата във формални термини, защото ако искаме да напишем конкретна програма, то трябва да ни е ясно какъв се очаква да бъде нейният вход и изход. Ще дадем и конкретен пример, който макар и прост е много показателен за разбирането на задачата. След тази подготовка ще започнем с описанието на конкретния алгоритъм и неговите модули. За да разберем света трябва да хванем зависимостите в този свят. Най-простите зависимости са зависимостите без памет. Това са зависимости от типа: „Ако виждам това и направя така ще се случи онова.“ Тоест, зависимостите без памет са връзката между последната стъпка и следващата. Ние ще обобщим и ще смятаме, че зависимостите без памет са връзката между последните няколко стъпки и следващата. Тези зависимости ще се представят като конюнкции на литерали и ще ги откриваме с груба сила като направим статистика на огромен брой конюнкции. Следващия въпрос, с който ще се занимаем това са зависимостите с крайна памет. Те ще бъдат представени като крайни автомати и тяхното откриване не може да стане с груба сила, а ще трябва много по-трики да се подходи.

Трябва ли да търсим зависимости с безкрайна памет. Ето пример за такава зависимост: „Ако до сега A се е случило повече пъти от B, то тогава нещо си.“ Хората не могат да откриват подобни зависимости и затова ще предположим, че ИИ също може да мине и без тях.

По-нататък в статията ще се занимаем с въпроса за абстрактните сигнали. Това е последното парче от пъзела, което липсваше за да решим задачата. Тези сигнали обективно съществуват в света, макар да не можем директно да ги наблюдаваме. Тяхната стойност може индиректно да бъде установена чрез експеримент. Важното за тези сигнали е, че те ни дават пълно описание на състоянието на света.

Следващата концепция, която се разглежда е концепцията за многоагентния свят. Оказва се, че единствената възможност да разберем един по-сложен свят е да отделим части от него в някакви черни кутии, които ще наречем агенти. Разбира се, ние няма да се занимаваме с това как тези агенти вътрешно са устроени, а само ще прогнозираме тяхното поведение. Например, за определен агент може да предполагаме, че той ни е противник и да очакваме от него винаги най-лошото.

Последния модул, който ще обсъдим в тази статия е модула за централизирано планиране. Планът, който този модул ще създаде представлява последователност от цели. Машината ще се насочи към първата цел, което ще е много по-лесно осъществимо от това да се насочи директно към крайната цел.

С какво тази статия се отличава от болшинството статии по ИИ. Ние разглеждаме ИИ като преобразовател (transducer), докато повечето ни колеги го разглеждат като функция. Разликите са много. Първо функцията няма памет и второ тя разпознава някаква входна информация едностъпково. По същество повечето статии в областта на ИИ разглеждат задачата как от множество от входни примери да се намери функция разпознавател. По същество това е задачата за апроксимация на функция. Пример за подобни статии това са работите в областта на невронните мрежи. Странно е как при такава проста задача резултатите са толкова незадоволителни. Основния проблем е, че разпознаването обикновено не е едностъпково. Ето ви пример: Виждате някой, който ви прилича на Пешо, но не сте сигурен. Махвате му с ръка и той ви маха. Значи е Пешо. Тоест, разпознаването е многостъпков процес свързан с провеждането на ред експерименти и затова трябва да търсим transducer, а не функция.

Заключение

Тази статия дава едно доста подробно описани е на алгоритъма на ИИ, но на нас не ни е нужен алгоритъм, а реално работеща програма. Каква е разликата между алгоритъм и програма? Разликата е като между архитектурния проект и реално построената сграда. Когато имаме един добър и изпълним архитектурен проект, по него можем да построим сграда. Работата може да е много, но тази работа е по-скоро техническа. Разбира се, между архитектурния проект и реалното

строителство стои инженерния план, защото всичко трябва добре да се сметне, иначе сградата ще рухне под напора на гравитацията.

Тук описанието на отделните модули е много грубо и незадоволително. Например модула, който търси зависимости с крайна памет (т.е. крайни автомати, които да са адекватни спрямо света). В случая е важно, че знаем какво търсим и на теория можем да го намерим с груба сила. Да, но тук метода на грубата сила няма да помогне, защото възможните модели (крайни автомати) са твърде много. Тук ни е нужен много по-интелигентен метод за търсене. Идеите, които са дадени в тази статия биха могли да са полезни за решаването на задачата, но въпреки това създаването на този модул си остава сериозно предизвикателство и в никакъв случай не се свежда до рутинно програмиране. Освен това, този модул може да има и самостоятелно приложение като програма за намиране на зависимости, която ще е полезна при решението и на други задачи.

След казаното до тук може би е по-добре да не сравняваме тази статия с архитектурен проект на бъдещата сграда на ИИ, а по-скоро да я сравним с груба скица, небрежно надраскана на хартиена салфетка.

Какво е ИИ?

Първата дефиниция на ИИ е дадена от Алан Тюринг и тя е следната: „Ако зад завеса са скрити човек и машина и ако ние разговаряме с двамата и не можем да познаем, кой е човека и коя е машината, то тогава тази машина е Изкуствен Интелект.“

Има много критики към тази дефиниция. Някой казват, че била неформална или че била субективна, защото зависела от преценката на изпитващия. Тази дефиниция действително е неформална, но това не е никакъв проблем. Има и известен елемент на субективизъм, но и това не е проблем. Когато изпитваме студенти също има елемент на субективизъм. Въпреки това казваме, че добър студент е този който си взима изпитите и не се притесняваме от субективността на това определение за добър студент.

Друга критика към дефиницията на Тюринг е, че тя вдига летвата твърде високо. Някой хора смятат, че от ИИ трябва да искаме много по-малко. Проблемът на тези хора е, че те не вярват в съществуването на ИИ и затова искат дефиницията на ИИ да не дефинира това дето несъществува, а нещо друго, по-простичко което вече е измислено.

Всички тези критики са напълно неоснователни. В дефиницията на Тюринг има само един единствен проблем и той е, че тя дефинира нещо повече от ИИ.

Ако ви кажа: „Представете си бирена бутилка“, то вие ще си представите бутилка пълна с бира. Това е вашата дефиниция, но по-правилно би било да си представите

празна бутилка, защото пълната бутилка е нещо повече от това, което искаме да дефинираме.

Същият е проблемът с дефиницията на Тюринг. Там се дефинира ИИ плюс образование. ИИ без образование е като празната бирена бутилка, но както ние знаем как да напълним празната бутилка с бира, така знаем и как да образоваме необразования ИИ.

На нас ни е нужна дефиниция на ИИ, която да определя ИИ без образованието. Такава дефиниция аз предложих през 2000 година [1]. Тя гласи: „ИИ е тази програма, която в произволен свят би се справила не по зле от човек.“

Тази дефиниция не зависи от образованието, защото както човек може да се роди в различен свят и да получи различно образовани, така и програмата може да бъде стартирана в различен свят. Стартирането на програмата отговаря на раждането при човека.

За да формализираме тази дефиниция трябва да кажем какво е свят, кога един се справя по-добре от друг и най-трудното е да кажем колко високо ще вдигнем летвата. Тоест, да кажем колко умна трябва да е програмата. Ако кажем „не по-глупава от човек“, това е доста неформално. Най-малкото има по-умни и по-глупави хора, макар че разликата в интелигентността между най-умния и най-глупавия човек не е чак толкова голяма.

За да се избегне сравнението с човек в [2, 3] се въвежда понятието коефициент на интелигентност, което е едно число което може да бъде изчислено за всяка програма. За ИИ се примат тези програми чието IQ е над определена стойност. Колко е това ниво не можем да кажем точно, както не можем да кажем колко ще бъде минималния бал за прием в Университета догодина.

В тази статия няма да се занимаваме с въпроса „колко интелигентен трябва да е ИИ“, а ще опишем програма, която е достатъчно интелигентна.

Постановка на задачата

Предполагаме че имаме дискретно време и че моментите от времето са естествените числа. На всяка стъпка машината ще получава вход (информация) от света и ще дава изход (действие) обратно към света.

Ще предполагаме, че всяка стъпка се състои от два момента, понеже по-късно ще разгледаме едни крайни автомати, които ще щракат по два пъти на стъпка (веднъж когато се получи вход и веднъж когато излезе изход). Щракат, т.е. променят състоянието си.

Ще предполагаме, че входът се описва от две функции (**View** и **Score**), а изхода от една функция (**Action**). Ще предполагаме, че входът постъпва в четните моменти от времето и че изхода излиза в нечетните. Тоест, първите две функции ще са дефинирани само за четните числа, а функцията **Action** ще е дефинирана само за нечетните. Входът го разделихме на две функции, защото ще предполагаме, че стойността на **Score** има смисъл, който не зависи от конкретния свят, а е общ за всички светове. Обратното, смисъла на функцията **View** зависи изцяло от конкретния свят и нашата машина ще трябва да се ориентира в конкретния свят и да разбере какъв е смисъла на информацията, която получава от функцията **View**.

Какъв е смисъла на функцията **Score**. Възможните стойности на тази функция ще са константата **Nothing** или някое число. Целта на нашата машина ще бъде да събере максимално голямо средно аритметично от всичките резултати на функцията **Score**, които са различни от константата **Nothing**.

Ще предполагаме, че функцията **View** връща **n** мерен вектор от скалари, функцията **Action**, съответно **m** мерен вектор от скалари, а **Score** връща едни скалар (**Nothing** или число).

В [1] предполагахме, че входът и изхода са булеви вектори, защото по този начин може да се кодира произволна крайна информация. По-късно решихме, че е добре да се избегне излишното кодиране, защото целта е да разберем света, а едно допълнително кодиране може да направи света още по-трудно разбираем. Сега предполагаме, че скаларите са крайни функции. Дори бихме могли да обобщим и да предполагаме, че стойността на скаларите може да бъде дори и цяло или реално число, но в тази статия няма да правим това обобщение и ще се ограничим с крайните скалари.

Произволният свят се състои от множество от вътрешни състояния **S**, (едно от които $\mathbf{s_0}$ е начално) и функциите **World**, **View** и **Score**. Функцията **World** ни казва какво ще е следващото състояние на света на следващата стъпка.

$$\mathbf{s_{i+2} = World(s_i, a_{i+1})}$$

Тук $\mathbf{a_{i+1}}$ е вектора, който се връща от **Action(i+1)**. Функцията **Action** не е част от света, нейните стойности са действията на нашата машина. Функцията **World** е дефинирана само за четни стойности на **i**, защото една стъпка се състои от два момента.

Функциите **View** и **Score** взимат за стойност $\mathbf{s_i}$ и връщат съответно **n** мерен вектор първата и едномерен вектор втората.

Ще предполагаме, че функцията **World** не е детерминирана и че при определен аргумент тя може да върне различни състояния, всяко от които си има някаква ненулева вероятност да бъде върнато. В частният случай, когато **World** е

детерминирана при определен аргумент тя ще връща точно едно определено състояние и ще го връща с вероятност единица.

Некоректни ходове

Ще предполагаме още, че функцията **World** не е тотална. Тоест, че не всеки ход е възможен. При определено състояние едни ходове ще са коректни, а други няма да са коректни. Добре е да предполагаме, че за всяко състояние има поне по един коректен ход. Тоест, че не съществува тупик, където машината да се забие и повече да не може да излезе. Ако съществуват такива тупици, то спокойно може ги асоцииираме с края на живота (няма следващ ход). Когато живота се окаже по-къс ще го оценяваме до там докъдето е стигнал. Естествена евристика за машината ще е да избягва подобни тупици и въобще да се стреми към ситуации, където има повече възможни ходове. Подобен инстинкт има и при хората. Те не обичат тесните затворени пространства, не обичат да са с вързани ръце и т.н.

Ще разрешаваме на машината да опитва некоректни ходове и няма да я наказваме за това. Нормално е некоректните ходове да се използват за събиране на информация. Например хората в тъмното опипват стените, за да разберат къде се намират. Човешката ръка не може да мине през стената и затова опипването на стената може да се приеме за некоректен ход.

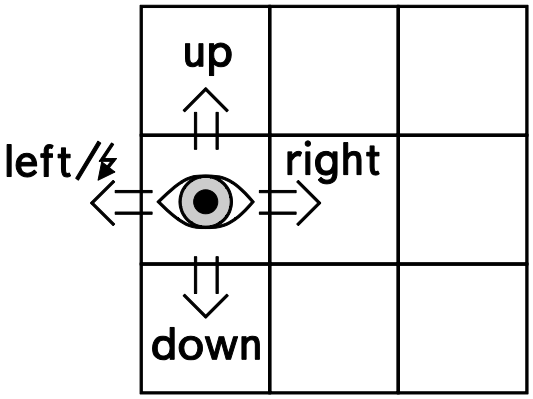
Некоректните ходове няма да променят състоянието на света и няма да увеличават брояча на стъпките **t**. Тоест, опитвайки некоректен ход машината нищо няма да загуби. Нищо няма и да спечели, освен това, че ще се сдобие с информацията, че този ход е некоректен.

Конкретен пример

За да се измъкнем от тресавището на абстракцията ще вземем един конкретен пример. Нека разгледаме света, където машината играе играта морски шах (Tic-Tac-Toe) .

Живота на машината трябва да е достатъчно дълъг, за да има тя време да се научи да играе. Затова ще предполагаме, че не се играе само една партия, а след края на всяка партия таблото се изчиства и започва нова партия.

Машината няма да вижда цялото табло на играта, а ще вижда само едно от квадратчетата. Тоест, ще има едно око, което ще може да се движи по таблото и машината ще вижда само това квадратче, върху което е окото.



Действието на машината ще се състои от тримерен вектор състоящ се от:
<движение по хоризонтала, движение по вертикала, поставяне на кръстче>

Всяка една от тези три координати може да взима по една от следните стойности:
движение по хоризонтала ∈ {на ляво, на дясно, на място}
движение по вертикала ∈ {на горе, на долу, на място}
поставяне на кръстче ∈ {постави кръстче, не поставяй}

Тоест, окото ще може да се движи във всички възможни посоки, дори и по диагонал (но само с една стъпка). Машината ще може да поставя и кръстче, но само в квадратчето, на което с позиционирано окото.

Нека отбележим, че далеч не всички ходове ще са коректни. Например, когато сме в лявата колона, хода на ляво ще е некоректен (без значение от другите две координати на изхода). Ходът постави кръстче също ще е некоректен, освен в случая, когато окото се намира върху празно квадратче. Тук некоректните ходове ще носят ценна информация. Например ще познаем, че сме в лявата колона по това, че хода на ляво е некоректен.

Функциите **View** и **Score** ще връщат едномерни вектори
View(s_i) ∈ {кръстче, кръгче, празно}
Score(s_i) ∈ {победа, загуба, реми, **Nothing**}

Тъй като казахме, че стойностите на **Score** трябва да са числа ще заменим победа, загуба, реми с числата 1, -1, 0, но ще имаме едно наум какъв е смисъла на тези числа.

Вътрешното състояние $\mathbf{s_i}$ на този свят ще се състои от едно табло 3x3 където има 9 квадратчета, всяко от които може да взима три възможни стойности плюс координатите на окото плюс още резултата на **Score** за това състояние. Последното се налага да го прибавим, защото когато завърши играта таблото се изчиства и е празно, а стойността на **Score** се определя от предишното състояние на таблото, последния ход на машината и евентуално от хода на въображаемия противник.

Какво се случва когато машината играе кръстче. Тогава въображаемия противник отговаря като слага кръгче на едно от празните квадратчета. Ще предполагаме, че това се случва веднага, още на същата стъпка и че още на следващата стъпка може да видим сложеното от въображаемия противник кръгче (стига да сме преместили окото до съответното квадратче). В случаите когато играта свършва след хода на машината или след хода на въображаемия противник, тогава таблото се изчиства и **Score** ни връща 1, -1 или 0 в зависимост от това как е свършила играта.

Това сякаш е напълно достатъчно за описанието на света, в който машината играе играта морски шах. Този свят изглежда абсолютно елементарен и има голям шанс машината да успее да го разбере напълно. Все пак, пропускаме нещо много важно и това е въображаемия противник. Той е част от света, защото вътре в света има някой, който играе срещу машината. Този някой е най-сложната и най-трудно разбираемата част от света.

Представете си, че въображаемият противник е професор по математика или влюбена студентка. Предполага се, че действията на професора ще са доста логични и че няма да е трудно да ги разберем и да предскажем следващият му ход. Много по-трудно ще е с влюбената студентка, защото тя ще играе нелогично и непредсказуемо. По надолу ще видим, че можем да извадим въображаемия противник от модела на света и по този начин модела силно ще се опрости.

Заключението е: Макар че светът на морския шах е съвсем прост, той има всички елементи, които са важни за програмата Изкуствен Интелект. Ако нашата програма успее да се справи с този свят, то има голям шанс тя да успее да се справи и в произволен свят, което ще значи, че действително удовлетворява дефиницията за ИИ. Разбира се, не трябва да хитруваме. Нашата програма сама трябва да намери правилата на играта и да се научи да я играе без ние предварително да сме ги вградили по някакъв начин в програмата.

История

История ще наричаме стойностите на функциите **View**, **Score** и **Action** в интервала от **0** до **t** (от раждането до текущия момент). Освен тези три функции ще добавим още функцията **IncorrectMoves**, която за всяко четно **t** ще връща множеството на некоректните ходове, които машината е опитала неуспешно в момента **t**. **IncorrectMoves** ще връща множество, а не списък, защото няма никакво значение в какъв ред тези ходове са опитвани.

На базата на историята машината ще събира статистика и ще се опитва да разбере света. На базата на последните няколко стъпки от историята ще се определя истинността на твърденията без памет. На базата на цялата история ще се определя истинността на твърденията с крайна памет. Твърдения с безкрайна памет няма да разглеждаме.

Твърденията без памет са конюнкции от минали литерали. Минал литерал е такъв, чиято стойност вече знаем, а бъдещ е такъв чиято стойност тепърва ще научим.

Дефиниция: Всеки литерал ще има вида: $\mathbf{View_i(t+k)=constant}$, където $\mathbf{View_i}$ е **i**-тата координата на вектора **View**, **t** е текущият момент от времето, а **constant** е някаква константа. Тук на мястото на функцията **View** може да стои някоя от функциите **Action** и **Score**.

Дефиниция: В зависимост от стойността на **k** литералите ще ги наричаме минали или бъдещи. Литералите ще са минали, когато $\mathbf{k \leq 0}$. Ако $\mathbf{k \geq 1}$, то литералите ще ги наричаме бъдещи и те ще се отнасят за бъдещето.

Защо ще се занимаваме само с конюнкции и няма да разглеждаме по-сложни булеви функции? Защото всяка булева функция може да се представи като дизюнкция от конюнкции на литерали и по тази причина, ако изследваме поведението на конюнкциите, това ще ни даде поведението на всички булеви функции.

По принцип нас ни интересува само бъдещето, но единствения начин, за да разберем и да прогнозираме бъдещето е като анализираме миналото.

Няма да предполагаме, че машината помни цялата си история, защото за това би била нужна много памет и освен това би било доста трудоемко да се анализира цялата история, за да се извлече някаква информация.

Това, което ще се помни от историята е един малък буфер от последните **k** стъпки и голямо количество статистика за огромен брой твърдения без памет. Като казваме огромен брой имаме предвид милиони и милиарди. Аз направих простичък експеримент [4], където дължината на конюнкциите беше ограничена до четири-пет и въпреки това броят на аксиомите нарасна до няколко милиона. Казвам аксиоми, а не конюнкции, защото тогава се интересувах само от конюнкции, които винаги до момента са били лъжа (т.е. отрицанията им са аксиоми, поне до момента). Сега противоречивите конюнкции пак са най-интересните, но вече ще се интересуваме и от останалите.

Друго нещо което трябва да се помни от цялата история е текущото състояние на твърденията с крайна памет. Всяко едно от тези твърдения се определя от състоянието на някакъв краен автомат. Текущото състояние на автомата трябва да се следи и да се знае (помни).

Дефиниция: Сигнал ще наричаме произволна функция, чийто аргумент е момент от времето и стойността й е скалар.

По-нататък ще разширим множеството от литералите като ще добавим и други сигнали към координатите на функциите **View**, **Action** и **Score**. Текущото

състояние на всеки един от автоматите съответстващи на твърдение с крайна памет ще е такъв сигнал.

Друго възможно разширение на множеството на литералите е да добавим някакъв абстрактен сигнал. (Абстрактния сигнал се определя от експеримент. За тези сигнали ще стане дума по-надолу.) По-точно ще добавим вероятността експеримента свързан с този абстрактен сигнал да върне истина. Тази вероятност е някакво число между 0 и 1. Възможните стойности на тази вероятност може да са една, две, краен брой и дори безкраен брой. Ако възможната стойност е само една (т.е. вероятността експеримента да върне истина винаги е една и съща), то няма да добавяме такъв литерал, защото литерали които са константа не са ни интересни. Ако възможните стойности са безбройно много, може с известно закръгление да ги сведем до крайно много. Най-добре е ако възможните стойности са две и те са 1 и 0. Тогава ще имаме два вида състояния. Такива, в които експеримента със сигурност ще успее и такива, в които със сигурност няма да успее. В последния случай може да направим само един експеримент и да разберем дали състоянието е от първия или от втория вид.

Твърдения без памет

За всяко едно, от тези твърдения ще има брояч за това колко пъти то се е случило (конюнкцията е била истина). Ще се помнят още последните M1 момента, в които твърдението е било истина. Ще се помнят още M2 стойности отговарящи на това, колко пъти твърдението е било истина в някакъв конкретен интервал от време. Машината ще изследва M2 конкретни интервали от време и за всеки един от тези интервали и за всяко твърдение от статистиката ще знае колко пъти това твърдение е било истина през този интервал от време.

Изследваните интервали от време няма да зависят от твърденията и ще бъдат общи за всички твърдения.

Например вие може да помните колко пъти ви е звъннал телефона днес откакто сте се събудил, тази седмица от първия работен ден и тази година от първи януари. Не твърдя, че помните точно, но имате идея, че тази седмица телефона се скъса да звъни, а днес специално беше спокойно. Ако ви питам колко звъня телефона ви преди 20 дена, то ще ви е трудно да ми кажете, макар че може да се опитате да изведете тази информация като си спомните какъв ден беше тогава и да го свържете с други събития. Тоест, това не го помните, макар че може би можете да го изведете по някакъв начин.

Тези M2 конкретни интервала от време ще бъдат разделени от M2 конкретни момента. Тези моменти ще са нагъсто близо да настоящия момент и ще се разреждат когато се връщаме назад към момента 0 (раждането). Разбира се, броя на тези моменти е най-много M2 и с добавянето на нов момент към списъка, ще трябва да изхвърлим някой от предишните. Например, ако помните колко пъти е

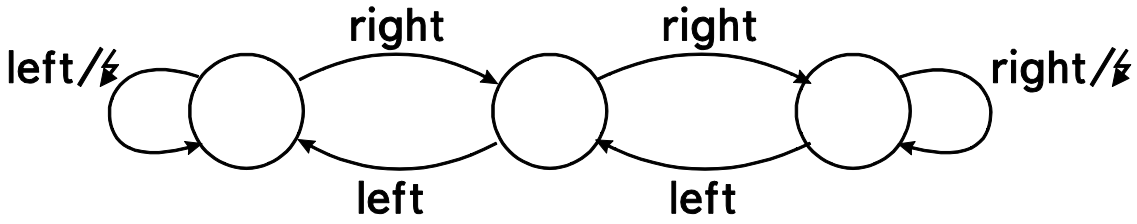
звънял телефона днес, от началото на седмицата и от началото на годината, тогава като се събудите отваряте нов брояч за телефонни позвънявания, но изхвърляте брояча от вчера сутрин, освен ако вчера не е било началото на седмицата, тогава ще запазите вчерашния момент и ще изхвърлите момента от миналия понеделник.

По-точно ще отворите нов брояч не след като се събудите, а след като ви звънне телефона (първото позвъняване след като сте се събудили, т.е. започнал е нов интервал от време). Ще увеличаваме брояча и ще отваряме нов брояч, само когато твърдението е истина. Няма да се занимаваме с твърдения, които не са истина в момента и така ще икономисаме компютърно време.

Трябва да дадем някакви конкретни стойности на константите M1 и M2. Например може те да са равни на 1000. Хубавото е, че увеличението на тези константи няма да забави нашата програма, а ще ни струва само разход на памет. За всяко твърдение, което се следи от статистиката ще са ни нужни M1+M2 цели числа. По този начин ние имаме добра памет за миналото. За всяко твърдение ние помним последните M1, момента в които то е било истина. Ако ни трябва по-стара информация, преди тези M1 момента, то ние помним през целия си живот колко пъти е било истина и дори можем да кажем колко пъти е било истина в отделни интервали от време, разбира се, не за всеки интервал, но все пак помним M2 конкретни интервала от време.

Най-важното, за което ще използваме твърденията без памет това е предсказване на бъдещето. Например, ако една конюнкция е противоречива, то нейното отрицание (то е аксиома) можем да го представим като импликация от конюнкцията на всички литерали без последния към отрицанието на последния. По този начин получаваме една импликация, с която можем да предсказваме бъдещето. Предполагаме, че импликацията е истина за всяка стъпка. Досега е била истина, предполагаме, че ще бъде истина и за в бъдеще. при положение, че е изпълнена

Твърдения с памет



Алгоритми

Абстрактни сигнали

Дотук се занимавахме само с миналото. Нека погледнем към бъдещето. При фиксиран брой стъпки аналог на твърденията без памет (конюнкции от минали литерали) са абстрактните сигнали (конюнкции от бъдещи литерали). Бъдещето е малко по-сложно от миналото, защото миналото е абсолютно детерминирано. Там ние знаем точно какъв ход сме избрали и каква информация сме получили. Всичко това е записано в историята. При бъдещето имаме един естествен недетерминизъм. Първо, не знаем какво точно ще видим, защото функцията **World** е недетерминирана. Второ, не знаем кой ход ще избере машината, защото тя е в правото си да избере който си ход иска. За да се справим с този недетерминизъм ще предполагаме, че стойността на конюнкцията не е вярно или невярно, а вярно с някаква вероятност и то при предположение, че хода на машината е точно който трябва.

Конюнкцията от бъдещи литерали ще бъде истина с някаква вероятност, при условие, че ходовете на машината са тези, които трябва (т.е. всички литерали свързани с **Action** да са истина). Всеки абстрактен сигнал можем да приемем, че е равен на вероятността определен експеримент да върне истина.

В примера с играта морски шах в лявата колона на таблото е в сила следната импликация.

horizontal(t)=left & vertical(t)=nowhere & put_cross(t)=no => bad_move

Тоест, в тази колона следващата конюнкция е невъзможна:

horizontal(t)=left & vertical(t)=nowhere & put_cross(t)=no

В другите колони конюнкцията е възможна с вероятност единица. Тоест, ако машината играе каквото трябва, то хода ще е коректен и другото също ще е изпълнено (то друго няма, защото всички литерали са свързани само с **Action**).

Тази конюнкция напълно определя хода на машината, но тя е твърде дълга (три литерала). Нека разгледаме конюнкцията с един литерал:

horizontal(t)=left

Тази конюнкция е невъзможна в лявата колона. В другите две колони тя е възможна с вероятност, която не можем да изчислим, защото не знаем кой точно ход ще играе машината. Например машината може да играе наляво и нагоре и хода да е некоректен, не защото не може наляво, а защото не може нагоре.

Конюнкциите от бъдещи литерали имат още един недостатък пред конюнкциите от минали литерали. Ние знаем стойността при миналите литерали, но не я знаем при

бъдещите и ни се налага да гадаем. Може на базата на някаква импликация и информацията, която имаме от миналото да предскажем стойността на абстрактния сигнал. Искаме на базата на тези абстрактни сигнали да правим статистика и по-късно да правим изводи. Лесно се прави статистика с миналите литерали. Броим, колко пъти е било истина и готово. При бъдещите литерали ние не знаем колко пъти е било истина. Далеч не всеки път сме правили нужния експеримент, а и да сме го направили от това че веднъж експеримента е успял или веднъж не е успял не можем да направим извода, че всеки път ще успява (или всеки път няма да успява).

Да вземем като пример експеримента „Ако пипна котлона ще се опаря“. Това е вярно когато котлона е горещ. Бихме искали да направим извода, че ако котлона е горещ можем да сварим кафе. Ако направим статистика на базата на моментите когато сме пипнали котлона и сме се опарили, то може да направим извода, че ако котлона е горещ, ще ни заболи ръката, а това не е защото котлона е горещ, а защото сме го пипнали. Затова статистика ще събираме, не на база на моментите, когато сме направили нужния експеримент, а на база на моментите, когато с достатъчно основание можем да вярваме, че котлона е бил горещ.

Тоест, ще се опитаме да предскажем с някаква вероятност, че котлона е горещ и на тази база да направи извода, че можем да сварим кафе. Ще се опитваме да познаем и кога котлона е бил горещ, за да съберем статистика от тези моменти. Тук предсказанието е различно, първо защото ни е нужна по-голяма увереност, за да не си развалим статистиката и второ защото може да използваме и следващи моменти. Например, пипнали сме котлона или някой друг го е пипнал. Това се е случило в следващ момент и не сме могли да го използваме, за да направим нужния извод в момента, но за статистиката е важно, че сме го разбрали, макар и със закъснение.

Как ще представим вътрешното състояние на света?

Бихме искали по някакъв начин да представим вътрешното състояние на света. Нас не ни интересуват недостижимите състояния на света и затова ще се погрижим само за тези, които са достижими тръгвайки от началното. Също така нас не ни интересуват еквивалентните състояния и ако имаме две неразличими състояния ще вземем само едно от тях. Затова ще предположим, че **S** се състои само от достижимите състояния и че е факторизирано по релацията еквивалентност.

Тъй като множеството **S** се състои от някакви произволни елементи, за които ние нищо не знаем, за нас би било достатъчно, ако намерим някое по-голямо множество, в което **S** да се влага инективно.

Дали би било достатъчно, за да опишем състоянието, ако вземем **n+1** орката състояща се от **View(s_i)** и **Score(s_i)**. Да, ако функцията **View** е инекция, то това би било достатъчно, но световете в които машината вижда всичко не са интересни. По-интересно е, когато машината вижда само част от света (т.е. когато **View** не е инекция).

Ако имаме две различими състояния **s'** и **s"**, то тогава тези състояния или са непосредствено различими, т.е. функциите **View** и **Score** ги различават, или има някакъв експеримент, който ги различава. Експеримента може да е с фиксиран брой стъпки или да е свързан с алгоритъм, чийто резултата да различава двете състояния.

Ето примери за експерименти с фиксиран брой стъпки (т.е. с една стъпка):
„Ако пипна котлона ще се опаря" или „Ако хвърля зарчето ще се падне шестица."

Ето пример за експеримент свързан с алгоритъм:
„Ако хвърлям зарчето последователно много пъти пет ще се падне преди шест"

Ето алгоритъма:
„Хвърлям докато не се падне нещо по-голямо от 4 и тогава ако е 5 то да, ако е 6 то не."

Експеримент с фиксиран брой стъпки ще наричаме една конюнкция от бъдещи литерали. По-нагоре беше обяснено какво е бъдещ и минал литерал.

Броят на стъпките на експеримента ще получим като вземем най-голямото **k** от конюнкцията, като го разделим на 2 и закръглим полученото нагоре.

Затворен (или напълно определен) експеримент ще наричаме такъв, в който участват всички литерали на **Action** (със стойности на **k** от 1 до най-голямото нечетно по-малко или равно на най-голямото **k** от конюнкцията). Тоест, действието на машината е напълно определено за стъпките на експеримента.

Отворен (или частично определен) експеримент ще наричаме такъв, който не е затворен. Т.е. действието на машината е частично определено. На отворения експеримент може да гледаме като множество от затворени, защото липсващите литерали на **Action** могат да бъдат добавени по всички възможни начини.

За всяко състояние, ако проведем един затворен експеримент ще получим резултат „да" или „не" с определена вероятност за „да". Ако функцията **World** е детерминирана, то тогава тази вероятност ще е точно 0 или 1. Ако **World** не е детерминирана, то тази вероятност ще е някакво число в интервала [0, 1]. Тоест, всеки затворен експеримент ни дава една функция, която на всяко състояние ни дава вероятността експеримента да завърши с „да". По-нагоре вече обсъдихме колко възможни стойности може да има тази функция? Най-добре е ако възможните стойности са само 0 и 1. Например, ако пипнем котлона той или е горещ и се парим или е студен и не се парим. Най-лошо е, ако възможната стойност е само една. Например, ако хвърля зара ще се падне ли шестица? Отговорът е „да" с вероятност 1/6 и това е така за всички състояния на света. Разбира се може да има свят, в който заровете може да са криви и когато хвърляме криви зарове вероятността да е по-голяма от 1/6. Може да има свят където талисманите помагат

и ако носим любимия си талисман вероятността отново да се увеличи. Тоест, две състояния са различими с експеримент не само ако за едното експеримента казва „да“, а за другото казва „не“. Различими са още, ако за двете експеримента казва „да“ с различна вероятност. Например различно е дали хвърляме прави или криви зарове, защото вероятността да се падне шестица е различна.

Ако си мислим, че функцията **World** е детерминирана, то резултата от затворения експеримент ще е определен, но това няма да ни помогне много, защото ние няма да успеем да го предскажем точно и ще очакваме резултата да е „да“ с определена вероятност. Тоест, при двата възможни модела на света (детерминиран и недетерминиран) ще имаме очакване за резултат от експеримента с една и съща вероятност, защото фактите, на които се базира прогнозата ни, са едни и същи.

Ако добавим към **n+1** орката вероятностите за успех от всички възможни експерименти, то резултата ще е безкрайно дълга редица от числа, но тази редица ще описва напълно състоянието на света с точност до еквивалентност.

Няма как да добавим всички възможни експерименти. Също така, не е добре да се ограничаваме само със затворените експерименти. Затова ще добавяме и отворени експерименти. Те са по-къси и в някакъв смисъл са по-представителни. За вероятността на отворения експеримент ще разглеждаме само три възможни стойности: „твърдо да“, „твърдо не“ или „може би“. Ще предполагаме, че имат поне две стойности, тоест поне една от тях е нещо твърдо. Когато обединим множество от затворени експерименти ние не можем да кажем каква е вероятността на обединението, защото не знаем каква е вероятността на всяко от действията (това зависи от машината, а на нея какво ще й хрумне, никой не знае). Случая, в който можем да кажем, е когато всички експерименти дават „твърдо да“. Тогава и обединението им е „твърдо да“. Аналогично, ако всички са „твърдо не“. Във всички останали случай стойността на обединението е „може би“.

В нашия пример с морския шах, ако окото се намира в лявата колона, то хода наляво е некоректен (т.е. невъзможен). Тоест, експеримента състоящ се само от един литерал връща „твърдо не“, когато сме в лявата колона и „може би“, ако сме в средната или дясната колона.

Защо абстрактните сигнали са важни?

Абстрактните сигнали са много важни за разбирането на света. Когато искаме да прескочим локва е много по-лесно, ако по средата на локвата има камък, на който можем да стъпим. Например ние намираме някаква зависимост според която котлона е горещ и зависимостта, според която, ако котлона е горещ, то можем да сварим кафе. Тогава на базата на тези две стъпки ние правим извода, че можем да сварим кафе. Бихме ли могли да намерим едно директно правило, че при тези обстоятелства бихме могли да сварим кафе? Да, но това директно правило би било много по-дълго и по-трудно откриваемо. Освен това, много трудно ще съберем

статистиката за това директно правило, защото неговата предпоставка почти никога няма да се е случила. Много по-лесно ще стане, ако е на две стъпки. Имаме някакви правила, които ни казват, че котлона е горещ. Всяко от тези правила е събрало достатъчно статистика за себе си. Имаме и някаква статистика, която ни казва, че ако котлона е горещ, то можем да сварим кафе. Това е на базата на всичките случай когато по някакви причини смятаме, че котлона е бил горещ. Тоест, тук имаме обобщение на много различни предпоставки. Ако не използваме подобни обобщения, то времето за учене на машината ще е огромно (т.е. броя стъпки преди да се научи ще е огромен). На теория можем да се обучим и без обобщения, но на практика това не може да стане, защото когато времето за учене клони към безкрайност, интелекта на машината клони към нула.

Важността на абстрактните сигнали за ИИ можем да я сравним с важността на Alpha–beta pruning при играта шах. Алгоритъма Min-Max може да работи и без тази хитрина, но тогава дълбочината на дървото намалява два пъти, а това не прави алгоритъма два пъти по-бавен, а да кажем милион пъти по-бавен. Подобно е и положението с абстрактните сигнали. Не само, че компютърното време се увеличава неимоверно, също толкова се увеличава и времето за обучение. Вие не бихте признали за ИИ програма която ще се научи да играе морски шах, но чак след изиграването на огромен брой ходове (например милион или милиард).

Многоагентен свят

Това което написахме досега изглежда достатъчно за разбирането на произволен сят. На теория това действително е достатъчно, но на практика не е. В примера с морски шах видяхме, че вътре в света има един въображаем противник, който е част от света. Ако искаме да опишем света заедно с въображаемия противник, то ще трябва състоянието на света да отразява състоянието на този въображаем противник. Дали е разсеян, уморен или ядосан. Това би направило описанието на света твърде сложно, затова бихме искали да извадим въображаемия противник и той вече да не е част от функцията **World**.

Старият вариант на **World**, който включваше въображаемия противник беше:

$$\mathbf{s_{i+2} = World(s_i, a_{i+1})}$$

Новия вариант ще бъде:

$$\mathbf{s_{i+2} = World\_2(s_i, a_{i+1}, opponent_{i+2})}$$

Тоест, ще имаме нова функция **World_2**, която ще взима като аргумент и действието на опонента (освен състоянието на света и действието на машината). Новата функция **World_2** ще е съвсем елементарна, защото няма да описва поведението на въображаемия противник, а само ще слага кръсчета и кръгчета на таблото на играта.

Идеята на многоагентното представяне на света е, че ние не сме сами на тази свят. Освен нас има още много други играчи (агенти). Това може да са хора, животни, богове или компютърни програми. Къде в нашия свят се проявяват действията на тези агенти. Щом някой друг прави нещо, то това нещо трябва да може да се появи по някакъв начин в света. Връзката между агентите и света са абстрактните сигнали. Има някакво състояние, което ние можем да проверим по някакъв начин. Стойността на това състояние се опитваме да я предскажем на базата на миналото или решаваме, че тя е случайна и ще се случи с определена вероятност. Вместо това може да предположим, че това състояние се определя по волята на някакъв друг неизвестен за нас играч. Този друг играч може да има властта да променя това състояние когато си пожелае, но по-добре да си мислим, че той е ограничен от някакви правила. Например в морския шах на празните квадратчета се появяват кръгчета. Ние имаме експеримент, с който можем да проверим какво става с празното квадратче. (Отиваме там и гледаме какво има в него.) Тук може да предположим, че въображаемия противник е ограничен и може само да сложи кръгче, но не може после сложеното да го махне.

За да разберем многоагентния свят ние трябва да имаме теория за другите агенти.

Първото нещо е да обединим група от проверими състояния в един агент. Например в морския шах някой слага кръгче в горния ляв ъгъл и някой слага кръгче в долни десен ъгъл. Добре би било да предположим, че това е един и същи агент. Обратното едно проверимо състояние може да бъде следствие на действията на различни агенти. Например, ако някой ви звъни по телефона. Това може да не е само един човек, а може да ви звънят много различни хора.

В теорията, която ще изградим трябва да има някакъв брой агенти и различните проверими състояния да ги свързваме с различните агенти. Например мама, която ни дава биберона или Господ, който отговаря за всичко.

Основно нещо за разбирането на другите агенти е да ги разделим на добри и лоши, т.е. на партньори и противници. Това ще ни помогне да предскажем действията им. Дали те ще се опитват да ни помогнат или да ни попречат. Това деление не трябва да го смятаме за постоянно. Партньора може да стане противник и обратното. В основата на нашата стратегия трябва да стои желанието да влезем в контакт с тези други агенти и да се разберем с тях по някакъв начин. Примери за подобен контакт са когато даваме подкуп, за да умилостивим някой чиновник или когато правим жертвоприношение, за да умилостивим някое божество.

Друг съществен момент за разбирането на другите агенти е да имаме теория за това кой какво вижда и кой какво знае. Има ли агента възможност да извърши определено действие. Както казахме има правила. Агента може в момента да не е там или в момента да не може да действа. Важно е умен или глупав е агента. Ние може да разчитаме, че той нещо ще го направи, но да знаем, че той е глупав и няма да се сети да го направи.

Това, което описахме, много прилича на начина, по който разсъждават хората. Всичко това биха били общи приказки, ако не бяхме казали какво е действието на агента, а то е промяната на някое проверимо състояние.

Централизирано планиране

Всичко, което разгледахме досега работи децентрализирано. Действително, търсенето на зависимости и построяването на модел на света може да стане децентрализирано, но когато машината иска да направи нещо на нея ще й е нужно да планира ходовете си. Да вземем нашия пример с играта морски шах. Там целта е да победим, но междина цел може да е да сложим кръстче в левия горен ъгъл на таблото. За да осъществи тази междинна цел трябва първо да преместим окото до левия горен ъгъл. Тоест, трябва да имаме планиращ модул, който да ни направи план за действие. Планът може да представлява редица от цели, които машината последователно трябва да достигне. Планът може и да е по-сложен и да не е редица, а да е сложен граф, защото може да има варианти.

В алгоритъма Min-Max няма централизирано планиране на действията. Там на всеки ход се изчислява кой е най-добрия ход за машината и този ход се играе. Този алгоритъм успешно играе играта шах, но не може да се справи с играта Го.

Представете си следната ситуация. Искате да заобиколите препятствие. Можете да тръгнете наляво, а можете да тръгнете и надясно. Ако имате планиране, ще изберете накъде да тръгнете и ще заобиколите. Ако нямате планиране, може да тръгнете наляво, да се откажете и да тръгнете надясно, после пак наляво и пак надясно. Подобно поведение би било доста странно, но по-лошото е, че ако не се планират междинни цели и ако на всеки ход се мисли в коя посока е генералната цел, то това би било твърде бавно и неефективно. Дори и алгоритъма Min-Max не се стреми към победа, а вместо това преследва повишаването на оценката на текущата позиция на таблото. Тоест, вградена е една междинна цел, която е лесно видима и е лесна за преследване. Крайната цел е много далеч и на практика е невидима.

Публикации:

[1] Dobrev D. D. AI - What is this. In: PC Magazine - Bulgaria, November'2000, pp.12-13 (www.dobrev.com/AI/definition.html).
[2] Dobrev D. D. Formal Definition of AI, International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.
[3] Dobrev D. D. Comparison between the two definitions of AI, January, 2013, arXiv:1302.0216 [cs.AI].

[4] Dobrev D. D. Generator of simple implications. (made in 2001 and published in 2008) (www.dobrev.com/AI/app4.html).

[5] Turing. Alan. Computing Machinery and Intelligence, In: Mind LIX (236): 433–460, October 1950.

[6] Balbiani Philippe, Gasquet Olivier, Schwarzentruber François Agents that look at one another. (2013) In: Logic Journal of the IGPL, vol. 21 (N. 3). pp. 438-467. ISSN 1367-0751

[7] Dochev D., G. Agre, R. Pavlov - User Authoring in Learning-by-Doing Situations. In: Rachev B., A. Smrikarov (Eds.) Proc. of CompSysTech 2011 , Vienna, June 2011, ACM ICPS Vol. 578, ACM PRESS, ISBN: 978-1-4503-0917-2, pp. 577-582.

[1] Dobrev D. D. A Definition of Artificial Intelligence. In: Mathematica Balkanica, New Series, Vol. 19, 2005, Fasc. 1-2, pp.67-74.
[2] Dobrev D. D. Testing AI in one Artificial World. Proceedings of XI International Conference "Knowledge-Dialogue-Solution", June 2005, Varna, Bulgaria, Vol.2, pp.461-464.
[3] Dobrev D. D. AI in Arbitrary World. Proceedings of the 5th Panhellenic Logic Symposium, July 2005, University of Athens, Athens, Greece, pp.62-67.
[4] Dobrev D. D. Formal Definition of AI, International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.
[5] Dobrev D. D. Injective Transliterations for Bulgarian and for Russian, (accepted for printing in International Journal on Information Theories and Applications).
[6] Dobrev D. D. Dobrev D. D. Parallel between definition of chess playing program and definition of AI. International Journal "Information Theories & Applications", vol.1, Number 2, 2007, pp.196-199.
[7] Dobrev D. D. Two fundamental problems connected with AI, June'2007 (represented at KDS 07, Volume 2, p.667)
[8] Dobrev D. D. Second Attempt to Build a Model of the Tic-Tac-Toe Game, June'2008 (represented at KDS 08), published in IBS ISC, Book 2, p.146
[9] Dobrev D. D. The Definition of AI in Terms of Multi Agent Systems, December'2008, arXiv:1210.0887 [cs.AI].
[10] L. Ivanov, D. Skordev, D. Dobrev. The new national standard for the Romanization of Bulgarian. Mathematica Balkanica, 24 (2010), Fasc. 1-2, 121-130.
[11] Dobrev D. D. Comparison between the two definitions of AI, January, 2013, arXiv:1302.0216 [cs.AI].
[12] Dobrev D. D. Giving the AI definition a form suitable for engineers, December, 2013, arXiv:1312.5713 [cs.AI].

Доклади на конференции:

3) Dobrev D. D. Use of Transliteration in the Transformation from Unicode to 8-bit code tables, TEI Members' Meeting workshop AZBUKY-NET, проведен в ИМИ през есента на 2005г.

4) Dobrev D. D. Formal Definition of AI and an Algorithm which Satisfies this Definition, Proceedings of XII International Conference "Knowledge-Dialogue-Solution", June 2006, Varna, Bulgaria, pp.230-237.
5) Dobrev D. D. Two fundamental problems connected with AI, Proceedings of XIII International Conference "Knowledge-Dialogue-Solution", June 2007, Varna, Bulgaria, Vol.2, pp.667-673.

2) Dobrev D. D. Parallel between definition of chess playing program and definition of AI. International Conference "Pioneers of Bulgarian Mathematics", July 8-10, 2006, Sofia, Bulgaria.
6) Dobrev D. D. The “sunshine” Method for Finding Finite Automata, New Trends in Mathematics and Informatics, July 2007, Sofia, Bulgaria, pp. 35-36.
7) Dobrev D. D. Comparison between the two definitions of AI, International Conference “Mathematics Days in Sofia”, July 2014, Sofia, Bulgaria, pp. 28-29.

Популярни:

[1] Dobrev D. D. AI - What is this. In: PC Magazine - Bulgaria, November'2000, pp.12-13 (in Bulgarian, also in [4] in English).
[2] Dobrev D. D. AI - How does it cope in an arbitrary world. In: PC Magazine - Bulgaria, February'2001, pp.12-13 (in Bulgarian, also in [4] in English).
[3] D. Dobrev. Кирилица или Методиевица - PC Magazine Bulgaria, стр. 12, 6/2001 г.
[4] D. Dobrev. Методиевицата се стандартизира - PC Magazine, Bulgaria, стр. 33, 2/2003.
[5] D. Dobrev. Искате ли да получавате пари вместо SPAM - PC Magazine, Bulgaria 4/2003
[6] D. Dobrev. Фонетична клавиатура за GSM - eWeek Bulgaria, стр. 9, 1/2004 г.
[7] D. Dobrev. Нови клавиатури - PC Magazine, Bulgaria 12/2004
[8] D. Dobrev. Телефонният чат - новата мания. PC Magazine Bulgaria 3/2006, pp.59-60.
[9] ?провери автора? D. Dobrev. Фонетична клавиатура за компютър и за GSM - PC Magazine Bulgaria, стр. 10, 11/2006 г.
[10] ст. н.с. Любомир Иванов, проф. Димитър Скордев, Димитър Добрев, Нужно ли е да стандартизираме клавиатурните подредби - Култура, 15 юни 2007 г. стр. 3.