Dialog state tracker - semestrální projekt

Vojtěch Hudeček

27. května 2016

1 Zadání úlohy, popis dat

Úvod

Zadáním je vytvořit model, který je schopen odhadovat z průběhu dialogu aktuální stav, to znamená, co uživatel v danou chvíli chce. Jedná se o soutěž pořádánou univerzitou v Cambridge, jíž se každoročně účastní desítky týmů.

Data

Poskytnutá data pochází z této soutěže, jedná se o záznamy dialogů mezi uživatelem a počítačem. Data pochází ze systému na zjišťování informací o restauracích. Sady dat obsahují informace o průběhu dialogu. Každý dialog se skládá z libovolného počtu kroků (*dialog turns*). Každý krok obsahuje uživatelovu promluvu (*usr*) a reakci systému (*sys*), společně s dalšími informacemi, především aktuálním stavem dialogu. Stav dialogu je reprezentován třemi sloty, jež mohou nabývat různých hodnot. První slot reprezentuje typ restaurace, druhý polohu a třetí cenovou kategorii. Výchozí hodonta slotů je *none*.

Předzpracování dat

Pro naše účely jsme data předzpracovali tak, že z každého kroku jsme vyrobili jeden řetězec spojením obou částí pomocí separátoru, vznikla tedy věta ve taru:*usr#sys*. Požadované výstupy jsou reprezentovány čísly, pro každou kombinaci zvlášť.

2 Model

Popis

Pro úlohu je vhodné použít model rekurentní neuronové sítě složené z GRU buněk, neboť data jsou sekvenční, je nutné brát do úvahy historii a navíc vstupy mají různou délku. Náš model je znázorněn na obrázku 1. Jedná se o klasický model rekurentní neuronové sítě rozvíjejíjící se v čase. Model sekvenčně bere jednotlivé vstupy, aktualizuje svůj skrytý stav a odhaduje výsledek, který je porovnán s požadovaným výstupem a na základě toho je určena hodnota chybové funkce. Díky tomu je umožněna modifikace parametrů pomocí algoritmu zpětného šíření.

Podrobnosti

- *Předkládání vstupu* - V našem řešení iterujeme přes jednotlivé dialogy, každý dialog je zpracován modelem. Dialog se skládá z určitého počtu tahů a každý tah ze slov. Postupně procházíme všechny tahy, model je sekvenčně zpracuje po slovech, spočítá chybu, uloží si stav, a zpracuje další tah.

- *Získání odhadu* - Výsledek je vypočten transformací skrytého stavu na požadovanou dimenzi pomocí projekční matice. Výstupy reprezentujeme pomocí *one-hot* vektorů. Následně použijeme funkci *softmax*, která z výstupu vytvoří pravděpodobnostní rozdělení. Predikovaná třída je ta s největší pravděpodobností.

- *Výpočet chyby* - Pravděpodobnostní rozdělení získané podle postupu uvedeného výše vyhodnotíme pomocí chybové funkce *cross entropy* a požadovaného výstupu.

- *Reprezentace dat, chybějící hodnoty* - Používáme metodu vektorových embeddingů, která každému slovu přiřadí mnohadimenzionální reálný vektor reprezentující toto slovo. S těmi pak pracuje síť. Hodnoty, které se nevyskytnou v datech jsou reprezentovány speciální konstantní hodnotou. Abychom model 'připravili' na tuto hodnotu, s nenulovou pravděpodobností se může vyskytnout i v trénovacích datech.

Implementace

Popsaný model jsme implementovali v jazyce Python 3 za pomocí knihoven *numpy* a *tensorflow*. Model podporuje dávkové zpracování. Po vytvoření výpočetního grafu jsou z načtených dat nahrávány jednotlivé dialogy a jejich kroky jsou postupně předkládány modelu pro učení. Chybovou funkci optimalizujeme pomocí metody *Adam*. Během implementace jsme museli překonat některé problémy, zejména rozdílnou délku vstupů. To řešíme maskováním dat na úrovni dialogů a předáváním délek výpočetnímu gafu na úrovni jednotlivých kroků.

3 Výsledky

Charakteristika dat, výsledky

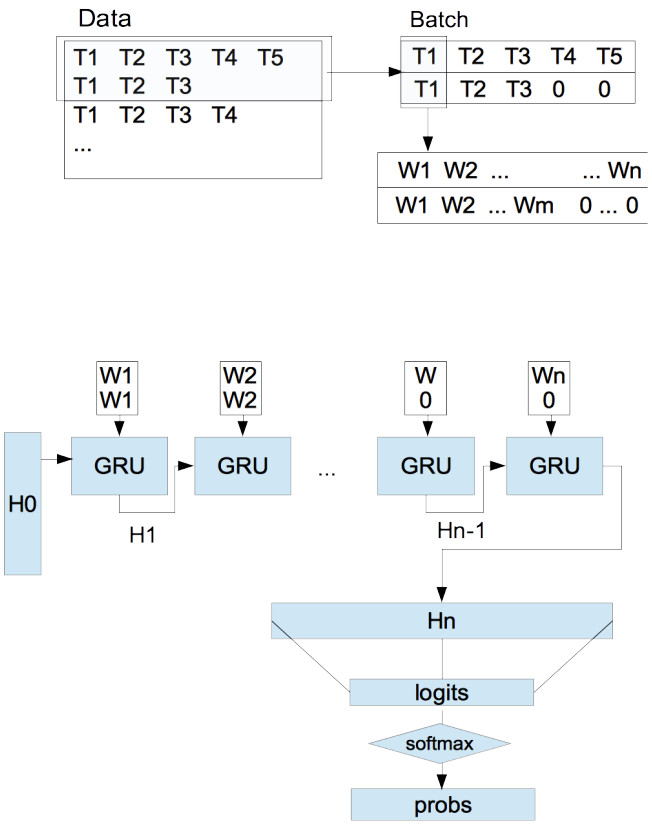
Jako klasifikační úloha je řešený problém poměrně obtížný. Je zde totiž velké množství tříd (895 pro trénovací množinu), z toho třída 0 je zastoupena v trénovací množině ve 22.104% případů. Počty tříd i rozdělení ve validační i testovací množině jsou velmi podobné. Nepoměr v zastoupení ilustruje i graf 2. Během učení náš model zlepšuje svou přesnost na testovací i trénovací množině. S dimenzí vnitřního stavu 150 a embeddingů 100, dosáhl model po 25 epochách přesnosti uvedené v tabulce 3. V tomto experimentu byla použita velikost batche 32. Když jsme ji snížili na 2, model dosahoval horší přesnost na trénovací množině, nicméně na testovací sadě se dostal po 15 epochách na 30%ní úspěšnost. V obou případech se model stále zlepšoval.

Vylepšení

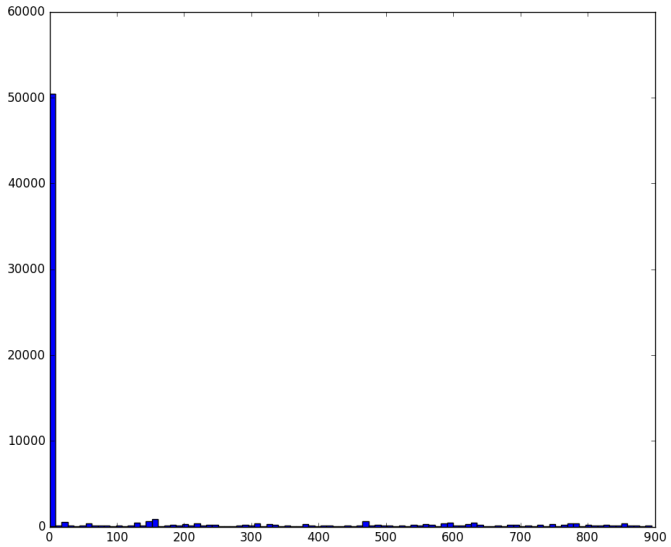
Provedené experimenty naznačují, že parametry které jsme nastavili nejsou pro tuto úlohu ideální. Bylo by vhodné věnovat čas nalezení optimálních parametrů, zejména dimenze skrytého stavu a vektorových embeddingů. Lepších výsledků by pravděpodobně bylo také možné dosáhnout tím, že by model ignoroval některé kroky, které mají jako výstup třídu 0. Stejně tak není ideální brát výstupy jako celek, alternativou je predikovat každý slot zvlášť. To může být přínosnější, neboť konkrétní kombinace se v datech vyskytují velmi řídce. Dále je zde prostor pro vylepšení modelu jako takového. Použití LSTM jako základní jednotky by intuitivně mělo dosáhnout lepších výsledků. Dále by mohlo být vhodné použít tzv. *bidirectional* model, který zpracovává vstupní sekvenci dvěma průchody, je tedy schopen využít i 'budoucího kontextu'. Stejně tak by implementace metody *dropout* měla dosáhnout lepší generalizace.

Závěr

Úspěšně jsme navrhli a implementovali představený model. Navzdory tomu, že výsledky nejsou ohromující, model se stabilně učí a je zde potenciál pro zlepšení.



Obrázek 1: Schéma modelu



Obrázek 2: Histogram zastoupení tříd, ilustrativní - šířka sloupce neodpovídá jedné třídě

train	valid	test
0.8031	0.2698	0.2473

Obrázek 3: Úspěšnosti modelu