Povećanje upotrebljivosti analitičkog rešenja u vidu eksponiranja Najm modela kao RESTful servisa

P. Prvulović, D. Vujošević

***Sadržaj* — Najm, platforma za analitiku podataka, dostupna je kao korisnički softver otvorenog koda. Aplikacija za kreiranje je istovremeno i platforma za izvršavanje modela, pa diseminacija modela krajnjim korisnicima na ovaj način može biti nepraktična.**

Stoga smo razvili i u radu opisujemo jedan vid interfejsiranja Najm modela ka krajnjem korisniku kroz prilagođen korisnički interfejs. Koristeći PHP, model eksponiramo kao RESTful servis i na primeru pokazujemo mogućnost integracije Najm modela sa softverom koji ima veb interfejs.

***Ključne reči* — integracija, Najm, model, PHP, REST, servis**

I. UVOD

NAJM[1] platforma nudi dva vida izvršavanja modela: interaktivno izvršavanje u okviru radnog okruženja i izvršavanje unutar klaud okruženja. Interaktivni režim je pogodan u svakodnevnom radu, u situacijama kada se sa podacima radi u hodu, po potrebi. Klaud rešenje omogućava pristup modelu kroz veb interfejs, gde su krajnjem korisniku eksponirani samo ekranski elementi za unos parametara (tekstualna polja, prilog fajlova i dr.) i donekle popravlja performanse, ali je taj vid dostupan kroz komercijalnu licencu [1].

Kod interaktivnog pokretanja modela korisniku je moguće prikazati

Petar Prvulović, Visoka škola akademskih studija Dositej, Beograd, Srbija (petar.prvulovic@gmail.com).

Dr Dušan Vujošević, Računarski fakultet Univerziteta Union, Beograd, Srbija (dvujosevic8@raf.rs)

[1] KNIME – the Konstanz Informtion Miner

ekranski interfejs sa elementima za unos parametara, međutim model ostaje dostupan za izmenu i, uopšte, korisnik je svestan da barata sa softverom koji je puno kompleksniji od potreba koje možda ima. Ovo nam daje povoda da pokušamo integraciju Najm modela sa drugim softverom, pri čemu to može biti jednostavan softver koji služi samo kao maska za unos parametara i pokretanje Najm modela, a može biti i ekranski interfejs nekog drugog alata u poslovnom okruženju koji bi trebalo da se integriše sa Najm modelom tako da mu prenosi parametre i koristi rezultate izvršavanja, tako da korisnik ne mora da primeti da se u konkretnom scenariju uopšte nalazi, kao deo, Najm model.

U prvom delu rada objašnjavamo namenu platforme Najm. Ukazujemo na značaj razvoja alternativnih pristupa za korišćenje rešenja napravljenih u ovoj platformi.

Mogućnosti pokretanja Najm modela preko sistemske konzole analiziramo u drugom delu rada. Dajemo Najm model koji ćemo koristiti kao primer i razmatramo način prosleđivanja parametara iz konzolne naredbe u model.

U trećem delu postavljamo konstrukciju i PHP skript kojim ćemo eksponirati Najm model u skladu sa principima REST arhitekturnog stila, tj. kao tzv. RESTful servis [2]. Implementiramo prihvatatnje parametara i pokretanje modela i razmatramo načine za preuzimanje rezultata rada modela.

U četvrtom delu dajemo primer klijentske strane veb aplikacije koju smo napravili. Objašnjavamo kako smo je integrisali sa RESTful servisom iza koga se nalazi Najm model.

Rad zaključujemo diskusijom o performansama i ograničenjima predloženog pristupa. Razmatramo mogućnosti daljeg unapređenja i primene ovakvog vida integracije u praksi.

II. PLATFORMA NAJM I UPOTREBLJIVOST PROGRAMA RAZVIJENIH U NJOJ

Platforma Najm omogućava da se definišu i sprovode najrazličitije analize iz domena nauke o podacima. Razvijena za upotrebu u bioinformatici, ova platforma danas se koristi u širokom spektru primena. Njome se analiziraju rezultati eksperimenata u hemiji i molekularnoj biologiji, na primer onda kada se u mnoštvu dobijenih podataka žele uočiti obrasci koji se ponavljaju. Ona može biti podrška procesima rukovođenja, u kojima olakšava odlučivanje iz domena nabavki, finansija, proizvodnje, upravljanja ljudstvom, usluživanja klijenata, i drugih.

Program se u Najmu najčešće razvija tako što se u panel grafičkog

korisničkog okruženja prevlačenjem mišem prenose ikonice odabranih čvorova. *Čvor* je parče koda kojim se vrši određeni deo analize podataka. Svrstan je u odgovarajući tip čvorova u bogatoj biblioteci gotovih, implementiranih algoritama obrade podataka. Čvorovi se mogu podešavati unosom parametara. Međusobno se povezuju strelicama koje određuju redosled izvršavanja pojedinih koraka analize.

Analiza često počinje učitavanjem podataka. Učitavanje se obavlja u jednom čvoru ili u nizu čvorova kojima se obavljaju obrade tipa ekstrakcije, transformacije i punjenja podataka, iliti tzv. ETL obrade. Na ove čvorove nadovezuju se čvorovi kojima se spovodi analitika podataka. Analitika može podrazumevati deskriptivnu statistiku, statistiku zaključivanja, prediktivnu analitiku, rudarenje podataka, mašinsko učenje, analizu velikih podataka i druge tipove analiza. Na posletku, analiza se obično završava čvorom ili čvorovima koji omogućavaju pregled dobijenih rezultata, bilo tabelarni ili grafički.

Neke od tipičnih primena Najma su traženje korelacija, klasterovanje, klasifikacija i analiza teksta. Korelacije ukazuju na moguću povezanost između posmatranih pojava. Klasterima se stavke po sličnosti grupišu u skupove. Klasifikacija omogućava da se, na osnovu iskustva s prethodnim pojavama, predviđa ishod za nove pojave iste vrste. Tekstualna analiza olakšava i automatizuje snalaženje s velikim količinama teksta.

Program definisan prevlačenjem, setovanjem i povezivanjem u okviru panela grafičkog korisničkog interfejsa pokreće se u okviru istog tog interfejsa. Pošto se analize podataka, po pravilu, sprovode samo povremeno i samo od pojedinih, kompjuterski visoko pismenih korisnika, ovakvo pokretanje zadovoljava tipične analize podataka. Uprkos tome, ovakvo pokretanje ima i dva ozbiljna nedostatka.

Ono, s jedne strane, zahteva instaliranje platforme kod korisnika, što u organizacionom kontekstu može biti prepreka na čije se rešavanje čeka dugo ili se, čak, nikada ne prevazilazi. Moguće je da informatička služba nema kapaciteta da obavi instaliranje. Ili se želi uštedeti na troškovima, na primer u onim organizacijama čija poslovna politika nalaže obaveznu nabavku licencirane verzije softvera otvorenog koda i njenu instalaciju na lokalnim serverima. Ili se prosto propušta da se uvide pojedini potencijalni korisnici.

S druge strane, korišćenje programa u okviru platforme predstavlja izazov za sve one korisnike koji se pribojavaju novih aplikacija, analitike, statistike, poruka o izuzecima i svega onoga što okruženje može pred njih postaviti. Ovakvi korisnici izbegavaće da sprovode analitiku podataka, naročito ako je ona tek jedan od zadataka čije se sprovođenje od njih očekuje. Iskustvo sa

projekata u praksi pokazuje nam da većina potencijalnih korisnika analitičkih rešenja potpada pod ovu grupu. Razvoj informatike, međutim, neumitno oslobađa korisnike od jednostavnijih zadataka, a paralelno se od njih sve više očekuje da sprovode kompleksnije, kreativnije zadatke.

Usled svega navedenog, neophodno je raditi na približavanju analitike podataka što većem broju krajnjih korisnika. Povećanje mogućnosti pristupa analitici, iliti povećanje upotrebljivosti analitičkih rešenja, u našem istraživanju sprovodimo omogućavanjem netipičnih pokretanja programa razvijenih u Najmu. Proučavamo kako se ovi programi mogu pokretati preko sistemske konzole i, na osnovu toga, a što je posebno značajno sa stanovišta povećanja upotrebljivosti, kako se oni mogu eksponirati u vidu RESTful servisa.

III. POKRETANJE NAJM MODELA PREKO SISTEMSKE KONZOLE

Najm omogućava da se model pokrene preko konzolne naredbe, u tzv. *Headless* modu [3]. *Headless* mod pokreće kompletan softver bez grafičkog editora i automatski izvršava model. Moguće je pokrenuti radni model ili model upakovan u zip arhivu. U prvom slučaju, model se ekskluzivno zaključava, pa onemogućava korišćenje modela za više korisnika istovremeno. U drugom slučaju zip arhiva se raspakuje prilikom izvršenja u privremenu lokaciju i nema zaključavanja.

A. Pokretanje modela u Headless modu

Podrazumevano, Najm se na *Windows*-u instalira na C:\Program Files\KNIME, gde se nalazi knime.exe izvršni fajl. Model se tipično nalazi na drugoj lokaciji, pa ćemo ovde uzeti D:\workspace.

Direktno pozivanje knime.exe fajla nam nije od koristi jer se tako pozvan proces pokreće odvojeno od pozivaoca, pa nemamo indikaciju da li je izvršenje modela završeno. Da bismo primorali knime.exe da se izvršava kao child proces, koristićemo start /w sistemsku naredbu [4].

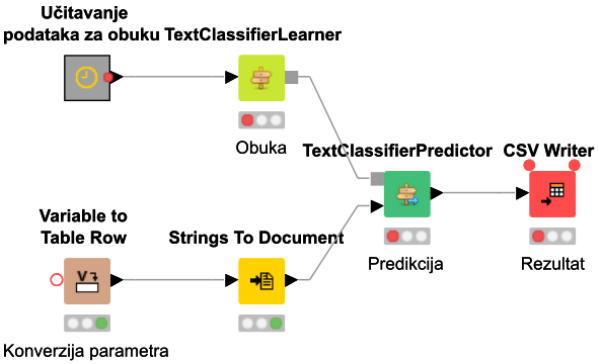
Konačni oblik konzolne naredbe za pokretanje modela upakovanog kao zip arhiva na lokaciji D:\workspace\model.zip je:

```
Start /w C:\Progra~1\KNIME\knime.exe –reset –nosplash –
nosave –application org.knime.product.KNIME_BATCH_APPLICATION
–workflowFile=”D:\workspace\model.zip”
```

Dokumentacija [3] opisuje parametre dostupne za pokretanje kroz konzolu. U ovoj naredbi, parametri imaju ulogu da se, redom, resetuje model, sakrije Najm logo prilikom pokretanja, spreči čuvanje novonastalih rezultata, pokrene aplikaciju u režimu paketne obrade i otvori model na datoj lokaciji.

B. *Primer Najm modela*

Kako bismo testirali postupak upotrebićemo model jednostavnog klasifikatora teksta. Implementacioni i logički detalji klasifikatora nisu suštinski bitni u ovom kontekstu, jer bi model mogao biti bilo koje namene. Pomenuti je izabran za primer isključivo iz razloga jer nije trivijalan primer već sadrži čvorove za obučavanje koji rade nad uzorkom od više od 10000 uzoraka, čime je bliži mogućoj realnoj primeni. Fokus stavljamo na one čvorove modela potrebne za eksponiranje modela kao RESTful veb servisa. U nastavku opisujemo model u meri potrebnoj da se stekne uvid u njegovu strukturu i rad, kako bi se videlo koja su mesta kritična za eksponiranje modela na način pogodan za dalju integaciju.



Sl. 1. Najm model kategorizatora upotrebljen kao test primer.

Model, prikazan na sl.1, obučavamo uzorkom naslova novinskih tekstova prikupljenih sa veb sajta www.b92.net u arhivi vesti u kategorijama *Vesti, Svet, Region, Društvo, Hronika, Kosovo* i *EU*. Prikupljanje kategorizovanih podataka sa veb sajta izvršeno je pomoću PHP skripta namenski napisanog za ekstrakciju podataka sa ovog sajta. Podaci su smešteni u CSV fajl, a kompletna procedura učitavanja podataka iz fajla i njihove pripreme za obučavanje prediktora grupisana je u meta-čvor sa nazivom „Učitavanje podataka za obuku“. Čvor učitava podatke iz unapred poznatog fajla, pa nema potrebe parametrizovati ga. Izlaz ovog meta-čvora su kategorisani naslovi vesti, u formatu pogodnom za čvor za obučavanje, koji je tipa *TextClassifierLearner*. Ovaj čvor obrađuje dati ulaz i formira pravila odlučivanja o načinu kategorisanja novih stringova [5].

Čvor *TextClassifierPredictor* [5] prima dva ulaza – pravila odlučivanja i podatak za kategorisanje. Kao izlaz daje predloženu kategoriju, koju dalje prosleđujemo čvoru *CSV Writer* pomoću kojeg je upisujemo u CSV fajl.

Podatak za kategorisanje je potrebno dobiti od korisnika modela. Ovaj podatak deklarisaćemo kao *String* promenljivu. Da bismo ga prosledili prediktoru potrebno je da ga konvertujemo u odgovarajući format pomoću čvorova *Variable to Table Row* i *Strings to Document*.

Unošenje stringa za kategorisanje bi se, u klasičnom načinu pokretanja modela, vršilo dodavanjem čvora *String input*, koji prikazuje ekranski element za unos teksta. Pošto želimo da model pokrećemo konzolnom naredbom, potrebno je da istim putem prosledimo parametre, pa ćemo umesto ovog čvora parametar deklarisati kao promenljivu na nivou modela [6]. Crveni kružić na čvoru *Variable to Table Row* naznačava da ovaj čvor koristi promenljivu, a u podešavanjima čvora zadajemo da je to promenljiva koja se zove *input*.

Rezultat se može prikazati na više načina. Pošto ga je potrebno dalje proslediti, prikazivanje rezultata u grafičkom obliku nije pogodno jer se ne može proslediti drugoj aplikaciji kao podatak. Skladištenje rezultata u csv fajl omogućava dalje prosleđivanje, budući da je CSV format prilično jednostavan za učitavanje i obradu. I ovde želimo da omogućimo zadavanje proizvoljnog imena izlaznog fajla, pa će se čvor *CSV Writer* vezati za promenljivu *filename*.

C. Prosleđivanje parametara modelu kroz konzolni poziv

Početne vrednosti promenljivih mogu se proslediti parametrom **–workflow.variable=*naziv,vrednost,tip*** [3]. U ovom, najužem, primeru imamo dve promenljive: *input* i *filename*, obe tipa *String*. Kao primer, uzećemo da želimo da kategorizujemo naslov „Anić: Nastavnici od dece prave pokvarene kalkulatore“ i da rezultat smestimo u fajl *D:\izlaz.csv*. Konačni oblik konzolnog poziva aplikacije je:

```
Start /w C:\Progra~1\KNIME\knime.exe –reset –nosplash –
nosave –application org.knime.product.KNIME_BATCH_APPLICATION
–workflowFile=”D:\workspace\model.zip”
-workflow.variable=input,”Anić:Nastavnici od dece prave
pokvarene kalkulatore”,String
-workflow.variable=filename,”D:\izlaz.csv”,String
```

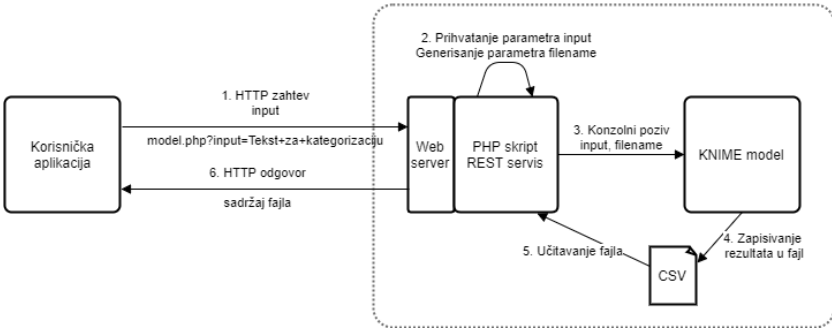
IV. EKSPONIRANJE MODELA KAO RESTFUL SERVISA

REST kao arhitekturni stil zahteva, između ostalog, da se resursi identifikuju kroz URI i da im se pristupa na uniforman način, pri čemu tok komunikacije ne treba da ima međustanja. U HTTP protokolu to se postiže upotrebom naredbi PUT, GET, POST i DELETE na logički konzistentan način, pa tako naredbu GET koristimo za dohvatanje resursa, pri čemu URI

može da sadrži i parametre, ukoliko oni jednoznačno određuju dobijeni rezultat. Rezultat je potrebno prikazati u takvom formatu da primalac može da ga dekoduje bez nužnog znanja o detaljima implementacije pošiljaoca, osim, naravno, formata u kome se odgovor dobija.

U našem slučaju rezultat HTTP GET zahteva će biti naziv prediktovane kategorije, za dati naslov vesti, pa je URI koji sadrži naslov vesti kao parametar u skladu sa REST metodologijom. Parametar *input* dobijen u URI-ju prosleđuje se Najm modelu, koji nakon izvršenja generiše CSV fajl. Sadržaj fajla konvertuje se i šalje kao odgovor u JSON formatu [7]. Tako, ciklus zahtev-odgovor nema međustanja, već se slanjem odgovora rad servisa završava, bez potrebe za dodatnim među-porukama. Konačno, JSON je standardizovan format tekstualnog predstavljanja strukturiranih podataka, te možemo reći da su REST principi zadovoljeni i nazvati ovo rešenje RESTful servisom.

U predloženoj konstrukciji RESTful servis se ponaša kao posrednik između korisnika (bio to čovek ili aplikacija) i Najm modela. Na sl.2 vidimo šematski prikaz elemenata konstrukcije i njihove međusobne interakcije i niz koraka u izvršenju jednog poziva Najm modela. Veb server, PHP skript i Najm se nalaze na istom računaru. Implementacija je pojednostavljena tako da je skript namenski povezan na konkretan model. Dalje unapređenje konstrukcije, tako da PHP skript bude ulazna tačka i na osnovu parametra poziva odgovarajući Najm model, je moguće i bilo bi strukturno bolje rešenje. Isto važi i za „ulepšavanje“ url-ova: url oblika host/knime/ime-modela bio bi više u duhu RESTful servisa.



Sl. 2. Elementi integracije Najm modela kao RESTful servisa.

Kako na sl.2 vidimo, tok izvršenja jednog poziva Najm modela je sledeći:

1. Korisnička aplikacija šalje HTTP zahtev u kome se nalazi parametar *input*, tj. tekst koji korisnik želi da kategorizuje

2. Veb server prihvata zahtev i prosleđuje ga PHP skriptu. PHP skript izdvaja parametar *input* i generiše vrednost za parametar *filename*, kako bi dobio unikatno ime fajla, čime se sprečava prepisivanje usled mogućeg istovremenog izvršavanja modela. Za unikatno ime fajla koristimo identifikator procesa PHP skripta.
3. PHP skript konstruiše konzolnu naredbu za pokretanje modela, koja u sebi sadrži i vrednosti parametara, inicira sistemski poziv te naredbe funkcijom *exec*[8] i ostaje u blokiranom stanju dok se model ne završi.
4. Najm model se pokreće, dobija parametre, na osnovu njih izvrši klasifikaciju i rezultat zapiše u fajl na lokaciji dobijenoj iz parametra.
5. Pošto se model završio, PHP skript nastavlja svoj rad i učitava fajl koji je Najm model generisao, na datoj lokaciji.
6. Sadržaj fajla se konvertuje u JSON format i ispisuje kao HTTP odgovor.

Kod PHP skripta dajemo u nastavku.

```
<?php
$id_procesa=getmypid();
$input=$_REQUEST['input'];
$filename="D:\\$id_procesa.csv";
//konstruisanje i izvrsenje naredbe
$naredba='start /w C:\Progra~1\KNIME\knime.exe -reset –nosave '
    .' -application org.knime.product.KNIME_BATCH_APPLICATION '
    .' -workflowFile="D:\workspace\model.zip" '
    .' -workflow.variable=filename,"'.$filename.'",String '
    .' -workflow.variable=input,"'.$input.'",String';
exec($naredba);
//priprema i slanje odgovora
$f = fopen($filename, "r");
$data = fgetcsv($f);
fclose($f);
$rezultat=new stdClass();
$rezultat->input=$data[0];
$rezultat->kategorija=$data[2];
echo json_encode($rezultat);
```

Skript se poziva URL-om oblika:

```
http://localhost/model.php?input=Anić:+Nastavnici+od+dece+prave+pokvar
ene+kalkulatore
```

a kao rezultat vraća JSON string oblika:

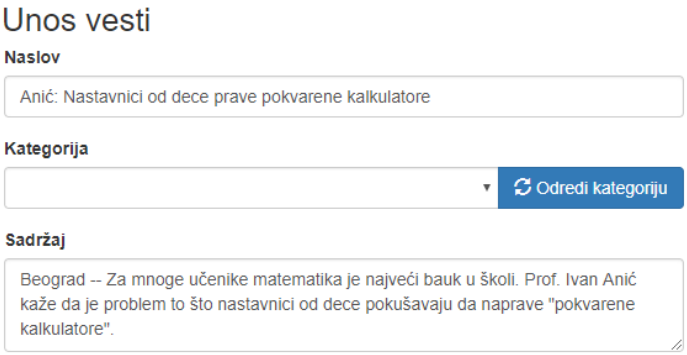
```
{“input”:”Anić: Nastavnici od dece prave pokvarene kalkulatore”,
”kategorija”:”Društvo”}
```

V. PRIMER KLIJENTSKE APLIKACIJE

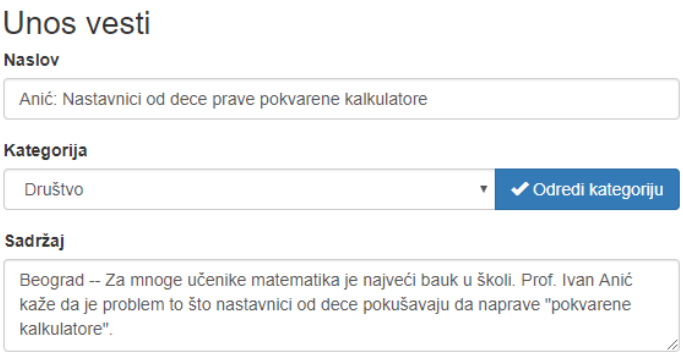
U nastavku dajemo kao primer interfejs veb aplikacije za unos vesti u bazu

podataka. Budući da je u pitanju veb aplikacija, interfejs je implementiran kroz HTML, uz upotrebu *Bootstrap* biblioteke [9] koja zahteva *jQuery* biblioteku [10], pa ćemo upotrebiti i dostupne *jQuery* funkcije kako bismo imali kompaktan i jasan kod.

Na sl.3 vidimo izgled obrasca za unos vesti. Obrazac sadrži polja *Naslov*, *Opis* i *Kategorija*. Uz polje kategorija je dugme *Odredi kategoriju*, koje korisnik, u slučaju da ne može da donese konačnu odluku, može upotrebiti kako bi dobio predlog kategorije na osnovu unetog sadržaja, tj. inicira se pozivanje Najm modela. Sl.4 prikazuje izgled obrasca nakon predlaganja kategorije.



Sl. 3. Izgled ekranskog interfejsa aplikacije za unos vesti u toku unošenja.



Sl. 4. Izgled ekranskog interfejsa aplikacije za unos vesti nakon dobijanja predložene kategorije.

Polje *Naslov* je *input* element, čiji je atribut id *naslov*. Polje *Kategorija* je *select* element koji u sebi sadrži *option* elemente za svaku kategoriju. Njegov

id je *kategorija*. Komandno dugme *Odredi kategoriju* je *button* element, čiji je id *kategorizuj*. Polje *Sadržaj* je *textarea* element, sa id-em *sadrzaj*. Id atribute pridružujemo kako bismo preko njih pristupali elementima.

Isečak HTML koda, u kome se definišu ova polja, dat je u nastavku.

```
<div class="form-group">
  <label>Naslov</label>
  <input type="text" id="naslov" class="form-control" />
</div>
<div class="form-group">
  <label>Kategorija</label>
  <div class="input-group">
    <select id="kategorija" class="form-control has-feedback">
      <option></option>
      <option>Društvo</option>
      <option>Vesti</option>
      <option>Svet</option>
      <option>Region</option>
      <option>Društvo</option>
      <option>Hronika</option>
      <option>Kosovo</option>
      <option>EU</option>
    </select>
    <span class="input-group-btn">
      <button class="btn btn-primary" type="button" id="kategorizuj">
        <span class="glyphicon glyphicon-refresh"></span>
        Odredi kategoriju
      </button>
    </span>
  </div>
</div>
<div class="form-group">
  <label>Sadržaj</label>
  <textarea class="form-control" id="sadrzaj" ></textarea>
</div>
```

Budući da PHP skript rezultate daje u JSON formatu, aplikacija koja inkorporira Najm model može prilično jednostavno da iskoristi te rezultate za dalju obradu. U primeru imamo HTML interfejs, pa ćemo komandnom dugmetu *Odredi kategoriju* pridružiti događaj pristika levog tastera miša. U obradi događaja potrebno je da dohvatimo vrednost polja *Naziv* i pošaljemo je kao neblokirajući, asinhroni HTTP zahtev PHP skriptu koji eksponira Najm model. Dobijeni odgovor je potrebno dekodovati iz JSON formata i polju *Kategorija* postaviti dobijenu vrednost.

U nastavku dajemo isečak JavaScript koda kojim se ovo postiže.

```
$("#kategorizuj").click(function(){
  //promena ikone kako bi se naznacio proces ucitavanja
  $("#kategorizuj span").addClass('glyphicon-refresh');
```

```
  $("#kategorizuj span").addClass('glyphicon-refresh-animate');
  $("#kategorizuj span").removeClass('glyphicon-ok');
  //slanje HTTP zahteva
  $.ajax({
    url: "http://localhost/model.php",
    data: {
      'input': $("#naslov").val()
    },
    success:function(odgovor){
      //definisanje nacina obrade dobijenog rezultata
      //promena ikone kako bi se naznacio kraj procesa
      $("#kategorizuj span").removeClass('glyphicon-refresh');
      $("#kategorizuj span").removeClass('glyphicon-refresh-animate');
      $("#kategorizuj span").addClass('glyphicon-ok');
      //dekodovanje odgovora iz JSON formata u JavaScript objekat
      var rezultat = JSON.parse(odgovor);
      //izbor predlozene kategorije
      $("#kategorija").val(rezultat.kategorija);
    }
  });
});
```

VI. ZAKLJUČAK

U radu smo pokazali vid integracije Najm modela bez upotrebe komercijalno dostupnih serverskih i klaud rešenja. Ovaj pristup, naravno, ima i prednosti i mana.

Bitna mana može biti vreme izvršavanja, pošto se pri svakom zahtevu pokreće celo Najm okruženje kako bi se model izvršio. S tim u vidu, ovaj pristup nije podesan za integraciju sa aplikacijama koje bi generisale veliki broj zahteva ka Najm modelu. Sa druge strane, ovo ne mora biti ograničenje ako je broj korisnika mali ili se model koristi dovoljno retko da ne uzrokuje zagušenje.

Ograničenje leži i u nemogućnosti korišćenja interaktivnih čvorova za prikaz rezultata, već se rezultati moraju uskladištiti u fajl kao tekst ili slika. Ovo je donekle rešivo prosleđivanjem podataka interaktivnim grafikonima na strani krajnje aplikacije, ali zahteva dodatni rad kako bi se grafikoni ugradili u aplikaciju. Ovo ne mora nužno biti problem, budući da Najm omogućava izradu prilagođenih grafikona u JavaScript-u, pa je u tom slučaju posao gotovo isti.

Sa druge strane, bitna prednost je mogućnost jednostavne zamene Najm modela drugom aplikacijom, bez sporednih efekata na aplikaciju koja koristi model. Ovo u praksi znači da Najm model može da se koristi u toku razvoja aplikacije, kako bi se model logički usavršio, a zatim jako jednostavno

zameni bržom implementacijom. Time se omogućava lako uključivanje stručnjaka za modelovanje podataka, koji ne moraju nužno biti tehnološki obučeni za okruženje u kojem se softver razvija, pa je ceo proces razvoja softvera bitno olakšan. U takvim situacijama, domenskom stručnjaku je dovoljno da zna Najm a da i dalje bude član tima i paralelno sa ostalima radi na razvoju aplikacije.

Drugi pozitivan scenario je mogućnost lakog unapređenja modela, u slučaju da model treba da evoluira. Domenski stručnjak koji koristi Najm može sa jako malo napora da dorađeni model postavi umesto postojećeg, a da pritom ne utiče na ostali softver koji taj model koristi.

Ovakva integracija omogućava dalju skalabilnost perfomansi u slučaju da korisnici žele da dodatno ulože u implementaciju modela koje bi na ovaj način razradili.

Autori smatraju da napori u ovom smeru mogu biti pozitivan pomak u daljem prihvatanju primene sistema za podršku u odlučivanju u poslovnom okruženju u Srbiji, budući da troškovi ovakve integracije mogu biti prihvatljivi i uprkos mogućim nedostacima, jer su svakako bitno manji nego kroz komercijalne licence i izradu namenskih modela, a dovoljno fleksibilni da budu od koristi i korisnicima koji tek počinju sa primenom i nemaju potpuno jasnu viziju kako bi mogli da iskoriste benefite ovog polja.

LITERATURA

[1] --, “KNIME Product Matrix”, *https://www.knime.com/products/product-matrix*

[2] X. Feng, J. Shen, Y. Fan, “REST: An Alternative to RPC for Web Services Architecture”, *Future Information Networks*, 2009. ICFIN 2009. First International Conference on

[3] --,“KNIME FAQ”, *https://www.knime.com/faq#q12*

[4] --,“Start, Command-line reference”, *https://technet.microsoft.com/en-us/library/bb491005.aspx*

[5] --,“Text Classifier”, *https://www.knime.com/book/text-classifier*

[6] --,“Flow Variables”, *https://www.knime.com/wiki/flow-variables*

[7] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014

[8] --,“exec”, PHP manual, *http://php.net/manual/en/function.exec.php*

[9] Reuven M. Lerner, “At the forge: twitter bootstrap”, *Linux Journal*, Volume 2012 Issue 218, June 2012

[10] Resig, John. "jquery: The write less, do more, javascript library." *http://jquery. com/*

ABSTRACT

The data analytics platform KNIME is available as an open-source software for end users. This workflow editor application is a workflow execution environment at the same time, which makes dissemination of workflows to end users somewhat impractical.

We have therefore developed an approach aiming to expose KNIME workflows to end users through a customized user interface. Using PHP, we expose a KNIME model as a RESTful service and propose a way to integrate it with other web-based software.

ENHANCING USABILITY OF ANALITICAL SOLUTION BY EXPOSING KNIME MODEL AS A RESTFUL SERVICE

P. Prvulovic, D. Vujosevic