Mesterséges Intelligenciát alkalmazó szövegbányászati eszközök készítése a *distiller* keretrendszer segítségével Jogi szövegek automatikus feldolgozása

Development of artificial intelligence-based text mining tools with the *distiller*- framework – in case of legal documents

OROSZ Tamás (PhD), CSÁNYI Gergely (PhD), NAGY Dániel (MSc)

MONTANA Tudásmenedzsment Kft. Budapest, Lágymányosi utca 11, H-1111
orosz.tamas@montana.hu, csanyi.gergely@montana.hu, nagy.daniel@montana.hu

Abstract

Machine learning projects do not fall under the traditional definition of software development. In addition to knowledge of software development and artificial intelligence methods, devops experience are also required for a successful project. These work management and deployment tasks generate significant overhead during the project. In this paper, we present an application built in an open source framework, through a legal text mining example, that not only enables agile task solving, facilitates teamwork among team members, but also makes the end-user deployment of the completed system solvable with a single command.

Keywords: machine learning, data mining, text mining, knowledge engineering

Kivonat

A gépi tanulást alkalmazó projektek nem tartoznak a hagyományos szoftverfejlesztés fogalomkörébe. Egy-egy projekt sikeréhez, a szoftverfejlesztési és mesterséges intelligencia módszerek ismeretén kívül rendszer üzembehelyezési és üzemeltetési tapasztalatokra is szükség van. Ezek, az üzembe helyezési, munkaszervezési feladatok jelentős overheadet generálnak a projekt megoldása során. Ebben a cikkben egy olyan, nyílt forráskódú keretrendszerben készült alkalmazást mutatunk be, egy jogi szövegbányászati példán keresztül, amely nem csak lehetővé teszi a feladatok agilis megoldását, elősegíti a csapattagok közti csoportmunkát, hanem az elkészült rendszer végfelhasználói üzembehelyezését is egyetlen paranccsal megoldhatóvá teszi.

Kulcsszavak: mesterséges intelligencia, adatbányászat, szövegbányászat, tudásmenedzsment

BEVEZETÉS

Napjainkban egyre több alkalmazás épül gépi tanulási módszerekre, hiszen a digitalizáció eredményeképpen nagy, strukturálatlan adathalmazok állnak elő, melyeknek a feldolgozása, egy adott dokumentum megtalálása rendkívül költséges feladat [1-3]. A gépi tanulással megoldható feladatok egyik alapköve az osztályozás, más néven klasszifikáció, amikor egy adathalmaz elemeit kategóriákba kell sorolni egy részben vagy egészben címkézett mintahalmaz alapján. Ilyen feladat például a kéretlen üzenetek szűrése a levelezőrendszerben [4], hitelesítendő pénzügyi dokumentumok szortírozása, vagy a cikkben bemutatott, jogi feladat, amihez több mint százezer, novella méretű jogi határozat osztályozása volt a feladat. Ennek a feladatnak a megoldásával, órákkal rövidíthető az ügyvédbojtárok feladata, akik egy adott ügyhöz tartozó precedens értékű határozatokat keresik ki az ügyvéd, ügyész vagy a bíró számára. Ennek a feladatnak az egzakt megoldása nem is biztosítható anélkül, hogy az egyes jogi kategóriába sorolható dokumentumokat leszámlálnánk.

A gépi tanulást alkalmazó projektek nem tartoznak a hagyományos szoftverfejlesztés fogalomkörébe. Az angol MLOps kifejezés [5] fejezi ki legjobban az ezekhez a projektekhez szükséges

szaktudást, amely szerint a szoftverfejlesztési ismeretek mellett szükségesek egyrészt a DevOps (rendszer üzembehelyezési) területét, másrészt a mesterséges intelligencia módszereket tudományos alapossággal ismerő szakértők. Bár napjainkban a szoftverfejlesztési projektek jól kialakult sztenderdek szerint vezethetők, a gépi tanulási feladatokat tartalmazó projektekben számos olyan fázis van, amikor egy-egy rosszul megválasztott fejlesztői eszköz vagy módszertan számottevő redundanciát hoz be a projektben elvégzett feladatok közé, ami végső soron felesleges költség. Ezek a feladatok jelenleg jobban hasonlítanak egy kutatási feladathoz, mint egy ipari szoftverfejlesztési feladathoz, így kevésbé jellemző a csoportmunka, ill. az egyes részfeladatok párhuzamosítása. Egy ilyen gépi tanulási projekt eredménye, rendszerint nem egy rendszerbe integrált ipari alkalmazás, hanem egy prezentáció, és/vagy az adott adathalmazból kinyert értékes adat.

A különböző AI modellek teszteléséből fakadó párhuzamosság, vagy a projekt során elkészülő modellek reprodukálhatósága ill. dokumentáltsága komoly üzleti probléma, amelyre jelenleg nincsen megfelelően kialakult, nyílt forrású, csapatmunkát, és ez által végsősoron egy projektet jól támogató eszköz.

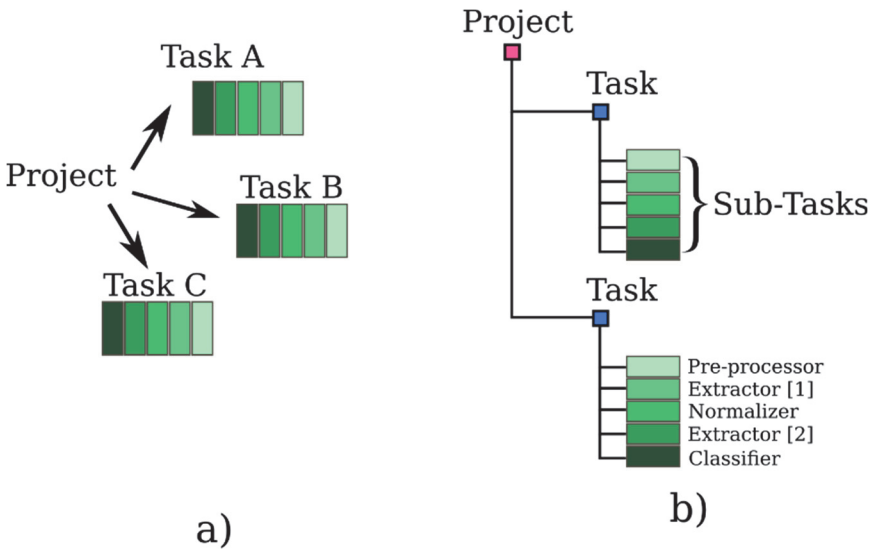
Szintén kulcsprobléma az elkészült mesterséges intelligencia modellek rendszerbeállítása: ez a folyamat az ipari projektek esetén jelentős overheadet generál. Ezt a feladatot oldja meg a distiller keretrendszer, amely képes a benne fejlesztett projekteket dockerizált formában, egy REST API-val ellátott, használatkész formában legenerálni. Ehhez egy parancs használata elég. A dockerizálás jelentősége abban áll, hogy rendszerarchitektúrától független, éveken keresztül használható alkalmazást kapunk.

DISTILLER

A *distiller* [6] projekt célja egy olyan keretrendszer fejlesztése, amely támogatja a csapatmunkát. Rengeteg repetitív feladat van egy mesterséges intelligencia alapú, szövegbányászati, képfelismerési vagy egyéb komplex mérnöki probléma megoldása során. Egy ilyen projekt életciklusának a négy nagyobb, jól elkülöníthető szakaszát ábrázolja az 1. ábra. A projekt első része jellemzően nem mérnöki vagy szoftverfejlesztési feladatokból áll, hanem magának a projekt tervezésével, a várt eredmények meghatározásával, a használandó modellek, módszerek kiválasztásával, ezek implementálásainak a tervezésével. A projekt leghosszabb és legidőigényesebb részét az adatok előkészítése jelenti, amely a gyakorlatban sokszor egy iteratív folyamat (data wrangling). Ebben a fázisban rengeteg adatból kell kiválogatni, kitisztítani, előfeldolgozni azokat az adatokat, amelyek a projekt során alkalmasak lesznek a modellek tanítására. Szintén probléma lehet, hogy nincs elegendő számú tanítóadat: ennek áthidalására a *distiller* saját, szöveges dokumentumokhoz használható augmentáló (*gyarapító*) modullal rendelkezik, amelynek segítségével a valódihoz hasonló, de szintetikus tanítóadat készíthető [7, 8]. Ezt követően a modell tréning és validáció már az a szakasz, ahol maga a modell kiválasztása, tanítása és validációja megtörténik.

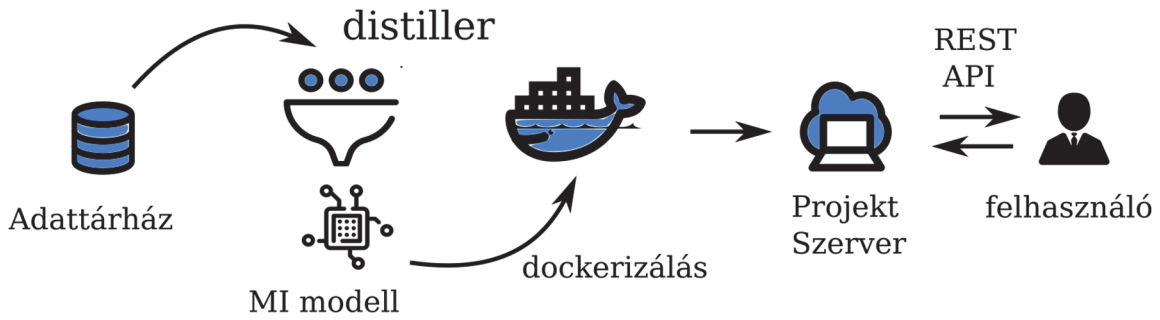
1. ábra. *Mesterséges Intelligenciára épülő projektek főbb megvalósítási fázisai*

Ennek a két szakasznak az elvégzéséhez rengeteg repetitív és egymásra épülő feladatot, például az adatok előfeldolgozását, különböző névelemek felismerését kell elvégezni, aminek egy adott Project – Task – Sub-Task osztálystruktúrába való leképezésével (2. b. ábra), egy adott projekten belül definiált feladatok (Task), illetve azon belül definiált részfeladatok (Sub-Task) felcserélhetővé és újrafelhasználhatóvá válnak. Így bizonyos feladatok párhuzamosan (2. a. ábra) is elvégezhetővé válnak. Erre jellemző példa, a Wolpert-féle “Nincs ingyen ebéd” következménye, hogy ugyanazt a feladatot többféle mesterséges intelligencia alapú módszerrel is meg kell oldani, hogy kiválasszuk az adott feladathoz legjobban illeszkedő eljárást [9]. Ezeket a feladatokat a *distiller* keretrendszeren keresztül több ember is végezheti egyszerre, illetve nagyon egyszerűen, egy-egy részfeladat cseréjével új algoritmusra cserélhető egy-egy része a feladatnak, úgy, hogy több projekt esetén nem igényel különösebb oktatást, illetve mélyebb előismereteket egy új kolléga számára. A keretrendszer akár kutatási projekteknél, vagy egymástól távol dolgozó kollégák esetén is megfelelő/standardizált kereteket biztosít a közös munkához.



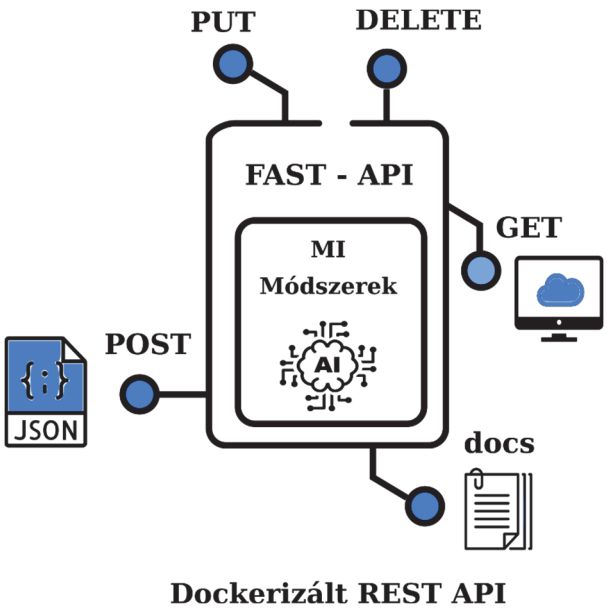
2. ábra. *Csoportmunka támogatása a distiller által Project – Task – Sub-Task hierarchia segítségével*

A mesterséges intelligenciát alkalmazó adatbányászati projektek negyedik, záró szakasza a projekt eredményeinek a vizualizálása, kiértékelése. Egy tudományos, vagy ipari kutatási feladat meg is áll ennél a lépésnél, azonban az ipari projektek megoldásánál egy ún. *production-ready* alkalmazás elkészítése a cél, amelynek a megfelelő becsomagolása, szállítása és üzembe helyezése komoly szoftverfejlesztési feladat. Egyes felmérések szerint, ennek a feladatnak a megoldása a projekt teljes idejének körülbelül a 10%-át [10] veszi igénybe. A *distiller* keretrendszerben készített projektek automatikusan szállíthatók, egy dockerizált [11] környezetbe csomagolva, ahol a végfelhasználó egy szerverként tudja futtatni a megvalósított alkalmazást.



3. ábra. *Mesterséges Intelligencia modellt tartalmazó applikáció fejlesztése és üzembe helyezése distiller keretrendszer segítségével*

A *distiller* projekt tartalmaz egy beépített *Server()* osztályt, amely a projektben megvalósított tudományos algoritmust egy dockerizált RESTful API-vá alakítja (4.ábra). Az alkalmazás egy *POST* hívásra küldött json fájlt a projekt osztály bemenetére rakja, majd a feldolgozott eredményekkel kiegészíti a megfelelő kulcsok alatt és a json fájlt visszaküldi. A szerver FastAPI [12] keretrendszer segítségével lett megvalósítva, amelybe az alkalmazott *pydantic* [13] modul segítségével json-schema validáció is beépíthető, támogatja az aszinkron kommunikációt. A projekt nyitóoldala tartalmazza az MkDocs-ban megírt dokumentációt, illetve a beépített teszt kliensen keresztül manuálisan is tesztelhető a dockerizált alkalmazás.

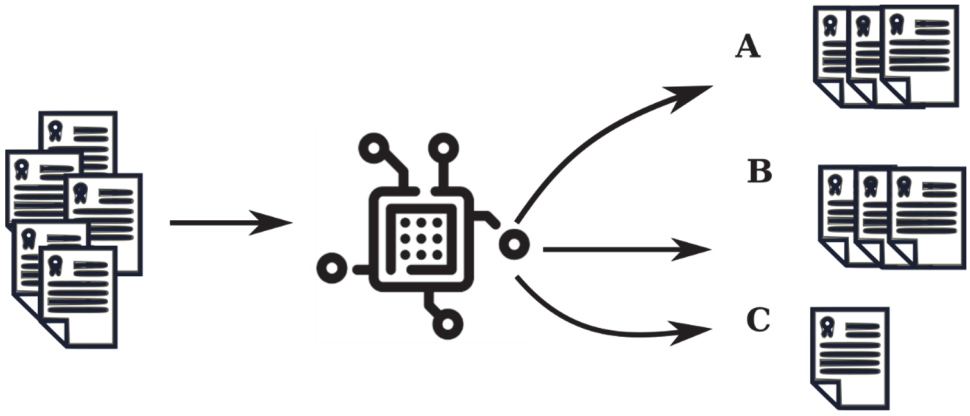


4. ábra. *A* distiller *projekt eredményeképpen automatikusan generált projekt szerver*

A dockerizálásnak köszönhetően, az elmentett modellek később, akár évek múlva is futtathatóak lesznek, különösebb rendszerismeret nélkül. Ennek a megoldásnak a következők miatt van jelentősége: a vállalati kompetencia fluktuációja, a vendor lock-in és a formátumok változása. A vállalati kompetencia nem egy állandó érték, hanem időről időre változik, ahogyan a munkavállalók feladatkört váltanak, vagy elhagyják a munkáltatójukat. Egy termelési láncba állított kód esetén nem engedhető meg, hogy azt kizárólag abban az esetben lehessen megbízhatóan működtetni, ha a teljes kifejlesztéshez szükséges kompetencia rendelkezésre áll. A vendor lock-in hasonló problémakör az előzőhöz, csak itt nem valaminek a hiánya, hanem épp egy bizonyos szolgáltatóhoz vagy gyártóhoz való kötöttség miatt állhat elő az a helyzet, hogy a vállalat nem ura a saját kódjának, ezért kiszolgáltatottá válik az üzletmenet terén. Végül a pedig a formátumok változásának azért van jelentősége, mert a mesterséges intelligencia módszereket tartalmazó Python könyvtárak, mint az *sklearn* [14], jellemzően *pickle* [15], vagy *joblib* [16] formátumban mentik el a modelleket. A Python *pickle* modulja az adott állapotában elmenti az összes osztályváltozójával a kódot, azonban, ha az osztály egyik adattagja megváltozik, akkor az a későbbi Python könyvtárral már nem használható újra. A *distiller*, a különálló, így önmagában is használható, GPL licenszes *sklearn2json* modult [17] használja a modellek mentésére, amely egy json fájlba menti az adatokat, így, ha változik az *sklearn* osztályváltozóinak az elnevezése, akkor ez az egy helyen karbantartott csomag képes lesz az új formátumra átalakítani régebbi modelleket, így újra felhasználhatóvá tenni azokat. Léteznek hasonló csomagok, pl. a kisebb észt startup által fejlesztett sklearn2pmml [18] csomag, ami azonban egy Java kódból generált Python API-val lenne használható, vagy az ONNX projekt [19], amely nagyvállalatok által fejlesztett ipari standarddá válhat. Ez egy közös fájlformátum, amely lehetővé teszi a mesterséges intelligencia fejlesztők számára, hogy a modelleket különböző keretrendszerekkel, és eszközökkel használhassák.

ESETTANULMÁNY

Nagymennyiségű, nyers szöveges adat feldolgozása általános probléma a mesterséges intelligenciát alkalmazó módszerek számára (5.ábra). A szöveges dokumentumok digitalizációja miatt rengeteg dokumentum keletkezik mind a különböző jogi, pénzügyi, vagy műszaki területeken, melyeknek a hitelesítése, keresése, feldolgozása komoly probléma.



5. ábra. *Nagymennyiségű szöveges dokumentum automatikus osztályozása a dockerizált REST API segítségével*

A cikkben bemutatott példa a jog területéről származik. A feladatunk az volt, hogy körülbelül 170 000 jogi határozatot kellett a tartalmuknak megfelelő jogi kategóriába egészen pontosan ügytárgy kategóriába besorolni. A szöveg elolvasása után eldönthető, hogy milyen jogi kategóriába sorolható a dokumentum.

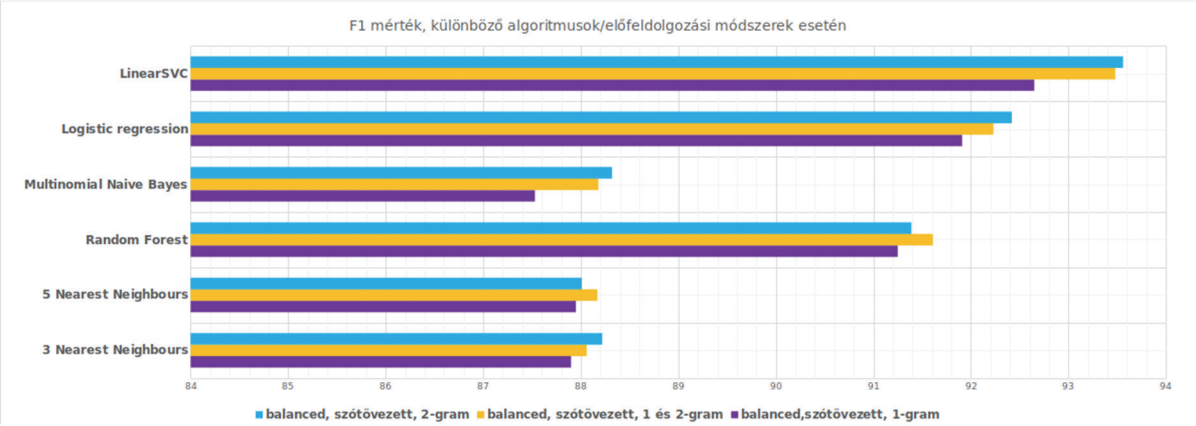
Erre azért van szükség, hogy a jogi adatbázisban könnyen kereshetővé váljanak a dokumentumok, mert a jogászi munka fontos része, hogy hasonló, precedens értékű ügyeket találjanak, ami alapján el tudják látni az ügyfél védelmét, vagy segíteni tudja a bíró munkáját a döntéshozásban. Ezt a munkát jellemzően a fiatal ügyvédbojtárok végzik, akik több órányi munkát eltöltenek egy-egy ügyhöz hasonló tényállást tartalmazó ügyek megtalálásával. Mivel a dokumentumok kategorizálatlanok, így nem tudjuk, hogy hány hasonló dokumentum van, és az is eldönthető, hogy az egyes esetekhez megtalálták-e az összes hasonló ügyet. Ennek a problémának a feloldására egy címkerendszert készítettünk, amely közel 170 kategóriába sorolja be az ügyeket, azzal a feltétellel, hogy az egyes ügyek több címkét is tartalmazhatnak, akár négyet is. Ez a kitétel nagymértékben megnövelte a probléma bonyolultságát.

A címkézési munka komoly szakértelmet igénylő, azonban rendkívül monoton munka, ugyanis egy – egy jogi határozat viszonylag hosszú, átlagosan 4-5 gépelt A4-es oldal hosszúságú. A validáció előállításához használt fájlok előállítása során azt tapasztaltuk, hogy kb. 100 határozat címkézhető fel egy munkanap alatt, ezt extrapolálva, a teljes adatcsomag kézi címkézése egy embernek körülbelül 8 évig tartana, és nem elhanyagolható az sem, hogy időközben évente tíz, tizenötezer új dokumentum generálódik.

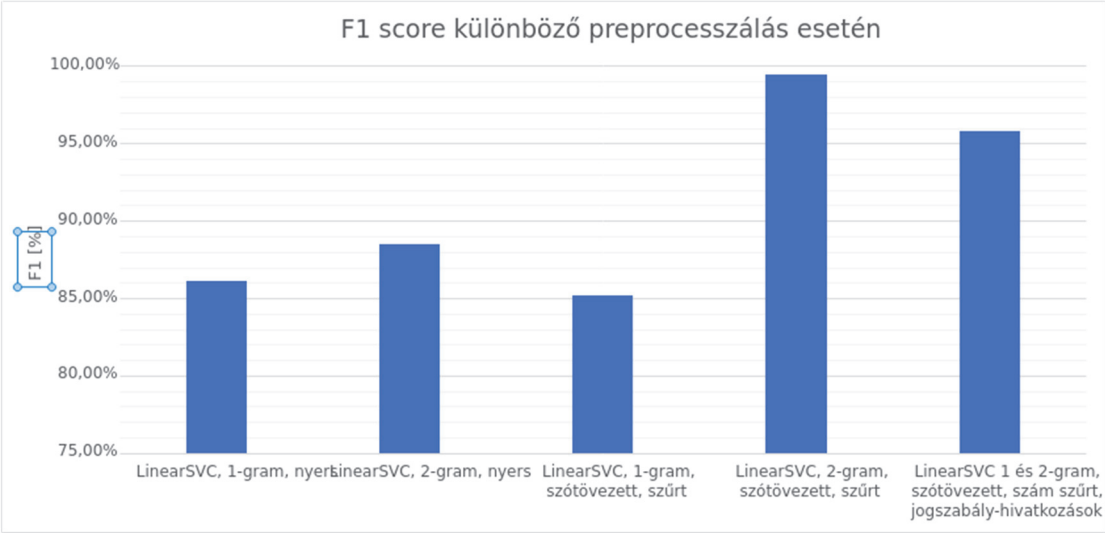
A következőkben bemutatott dockerizált mesterséges intelligenciára épülő alkalmazás segítségével nem csak a nagymennyiségű adat egyszeri leválogatását végeztük el, hanem készült egy production-ready dockerizált modell, amely az ügyfél rendszerébe beépülve automatikusan leválogatja az újonnan keletkező jogi dokumentumokat.

3.1 Jogi szövegek osztályozásához használt mesterséges modellek

A feladat a címkézési feladatokon belül a többcímkés osztályozási feladatok közé tartozik (multi-label classification), mert minden egyes dokumentumhoz egyszerre több címke is rendelhető. Ezt a feladatot bináris klasszifikálók használatával oldottuk meg, azaz minden egyes címkéhez egy önálló címkézőt készítettünk [20]. Azért döntöttünk emellett az egyszerű eljárás mellett, mert a 170 000 adat egyáltalán nem volt felcímkézve, reguláris kifejezésekkel és kézi ellenőrzéssel kellett az első tanítóadathalmazt feltanítanunk. Azért csak az elsőt, mert az így feltanított modellekkel további dokumentumokat szitáltunk ki a jogterület alá tartozó összes dokumentum halmazából, majd 2-3 iterációt követően eljutottunk azokhoz a címkéző algoritmusokhoz, amelyek bekerültek a leszállított algoritmusba.



6. ábra: *Különböző gépi tanulási modellek F1 mértéke 1, 2 illetve 1 és 2 gramok használatával vektorizáláskor*



7. ábra: *F1 mérték különböző előfeldolgozások esetén*

Bár manapság a transzformátor alapú megoldások (pl. BERT [21]) vagy a mélytanulás alapú megoldások (pl. [22]) általában jobb teljesítményt nyújtanak a hagyományos gépi tanulási módszerekhez képest, dokumentumosztályozás esetén a Support Vector Machines, Logistic Regression osztályozók egyszerű, de elég jó megoldásokat tudnak nyújtani a hosszú jogi dokumentumokra [23]. A tanítás során négy különböző gépi tanulási algoritmust teszteltünk és értékeltünk ki, nevezetesen a lineáris kernellel rendelkező Support Vector Machine (SVM), a Naive Bayes, a Logistic Regression, a Nearest neighbours és a Random Forest algoritmusokat. Ahol erre lehetőség volt, ott a balanced class weight beállítást használtuk, ami segítette az osztályozókat abban, hogy jobb teljesítményt érjenek el a kiegyensúlyozatlan adatkészleteken. A szövegek vektorizálására a jól ismert tf-idf vektorizálást használtuk, szóegyeseket, illetve szópárokat alkalmazva [24]. A modelleket kétszer megismételt 10-szeres keresztvalidálással értékeltük ki minden egyes tanítóhalmazon. A keresztvalidálással során a tanító és validáló halmazokat a pozitív-negatív minták arányát követő módon választottuk ki. Értékelési metrikaként az F_1 mértéket használtuk. Az osztályozók teljesítményét a "Munkaviszony megszüntetése" ügytárgyú dokumentumok esetén hasonlítottuk össze, szóegyesek, szópárok, illetve a kettő vegyített használatával (lásd 6. ábra). Az eredmények alapján a lineáris kernelű Support Vector Machine teljesített a legjobban A különböző előfeldolgozási lépések hatását is megvizsgáltuk, melyet a 7. ábra mutat be. Ezek a következő lépések voltak: szótövezés, írásjel és számok szűrése, jogszabály-hivatkozás kinyerés. A legjobban az a modell teljesített, amely csak 2-gramokat, tehát szópárokat tartalmazott, viszont a számok nem voltak kiszűrve. Mivel a szövegben szereplő számok alapvetően félre is vezethetik az algoritmust, így ezeket kiszűrve, és jogszabály-hivatkozásokat is felhasználva, valamint az összes szó használata helyett csak a 20 ezer legrelevánsabb megtartásával sikerült jóval kisebb modellel hasonló eredményt elérnünk.

KÖVETKEZTETÉSEK

A cikkben bemutatott nyílt forráskódú keretrendszer (*distiller*), lehetővé teszi, hogy az elkészített, akár többfajta mesterséges intelligencián alapuló módszerünket a végfelhasználók egy REST architektúrájú kliensen keresztül érjék el. A végfelhasználóhoz dockerizált formában juttatható el a csomag, melynek köszönhetően a bemutatott szövegek osztályozására szolgáló rendszer, a szoftverkörnyezet változásától függetlenül, hosszú távon is használható, egyszerűen telepíthető lesz. A MONTANA Tudásmenedzsment Kft-nél folytatott projektjeink során elsősorban a saját problémánkra igyekeztünk választ adni akkor, amikor kifejlesztettük a *distiller* Python csomagot. A jogi szövegek osztályozására használt rendszerben, többféle sekély tanuláson alapuló megoldást vizsgáltunk meg és építettünk be, melyek elegendő számú tanítóadat esetén 90% feletti eredményt produkálnak.

KÖSZÖNETNYILVÁNÍTÁS

A tanulmány a 2020-1.1.2-PIACI-KFI-2020-00049 számú projekt a Nemzeti Kutatási, Fejlesztési és Innovációs Hivatal támogatásával valósult meg, a 2020-1.1.2-PIACI KFI finanszírozási rendszer keretében.

IRODALMI HIVATKOZÁSOK

[1] Zhang, X., Luo, H., Fan, X., Xiang, W., Sun, Y., Xiao, Q., ... & Sun, J. Alignedreid: *Surpassing human-level performance in person re-identification.* arXiv preprint arXiv:1711.08184. (2017).
[2] Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C., & Ng, A. Y. *Cardiologist-level arrhythmia detection with convolutional neural networks.* arXiv preprint arXiv:1707.01836. (2017).
[3] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. *Dermatologist-level classification of skin cancer with deep neural networks.* Nature, 542(7639), 115-118. (2017).
[4] Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. *Survey of review spam detection using machine learning techniques.* Journal of Big Data, 2(1), 1-24. (2015).
[5] ML Ops Challenges, Solutions and Future Trends, https://towardsdatascience.com/ml-ops-challenges-solutions-and-future-trends-d2e59b74dc6b (2021.09.17)
[6] MONTANA Ltd. *Distiller framework.* (2021) https://bitbucket.org/montanatudasmenedzsmentkft/distiller/src/master/ (2021.09.17.)
[7] Csányi, G. M., Nagy, D., Vági, R., Vadász, J. P., & Orosz, T. *Challenges and Open Problems of Legal Document Anonymization. Symmetry*, *13*(8), 1490. (2021).
[8] Csányi, G., & Orosz, T. *Comparison of data augmentation methods for legal document classification. Acta Technica Jaurinensis*. (2021).
[9] Wolpert, D. H., & Macready, W. G. *No free lunch theorems for optimization.* IEEE transactions on evolutionary computation, 1(1), 67-82. (1997).
[10] Lawrence E. Hecht. Add It Up: How Long Does a Machine Learning Deployment Take?, https://thenewstack.io/add-it-up-how-long-does-a-machine-learning-deployment-take/ (2021.09.20.)
[11] Merkel, D. *Docker: lightweight linux containers for consistent development and deployment.* Linux Journal, 2014(239), 2. (2014).
[12] Ramírez, Sebastián. *FastAPI.* (2019). https://fastapi.tiangolo.com/ (2021.09.17.)
[13] Samuel Colvin. *Pydantic.* (2021) https://pydantic-docs.helpmanual.io/ (2021.09.17.)
[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. *Scikit-learn: Machine learning in Python.* The Journal of machine learning research, 12, 2825-2830. (2011).
[15] Van Rossum, G. *The Python Library Reference, release 3.8.2*, Python Software Foundation. 2020.
[16] Joblib Development Team. *Joblib: running Python functions as pipeline jobs*. 2020. https://joblib.readthedocs.io/ (2021.09.17.)
[17] MONTANA Ltd. Sklearn2json. (2021.) https://bitbucket.org/montanatudasmenedzsmentkft/sklearn2json/src/master/ (2021.09.17)
[18] Openscoring Ltd. Sklearn2PMML. https://github.com/jpmml/sklearn2pmml (2021.09.17.)
[19] Bai, Junjie and Lu, Fang and Zhang, Ke and others. *ONNX. Open Neural Network Exchange.* 2017, https://github.com/onnx/onnx. (2021.09.17.)
[20] Pereira, Rafael B., Alexandre Plastino, Bianca Zadrozny, and Luiz HC Merschmann. "Categorizing feature selection methods for multi-label classification." Artificial Intelligence Review 49, no. 1 (2018): 57-78.
[21] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. (2018).

[22] Wu, M., Liu, F., & Cohn, T. Evaluating the utility of hand-crafted features in sequence labelling. *arXiv preprint arXiv:1808.09075*. (2018).
[23] Suthaharan, Shan. "Support vector machine." In *Machine learning models and algorithms for big data classification*, pp. 207-235. Springer, Boston, MA, 2016.
[24] Luhn, H. P. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, *1*(4), 309-317. (1957).